NIST Special Publication 500-173

Guide to Data Administration

Bruce K. Rosen

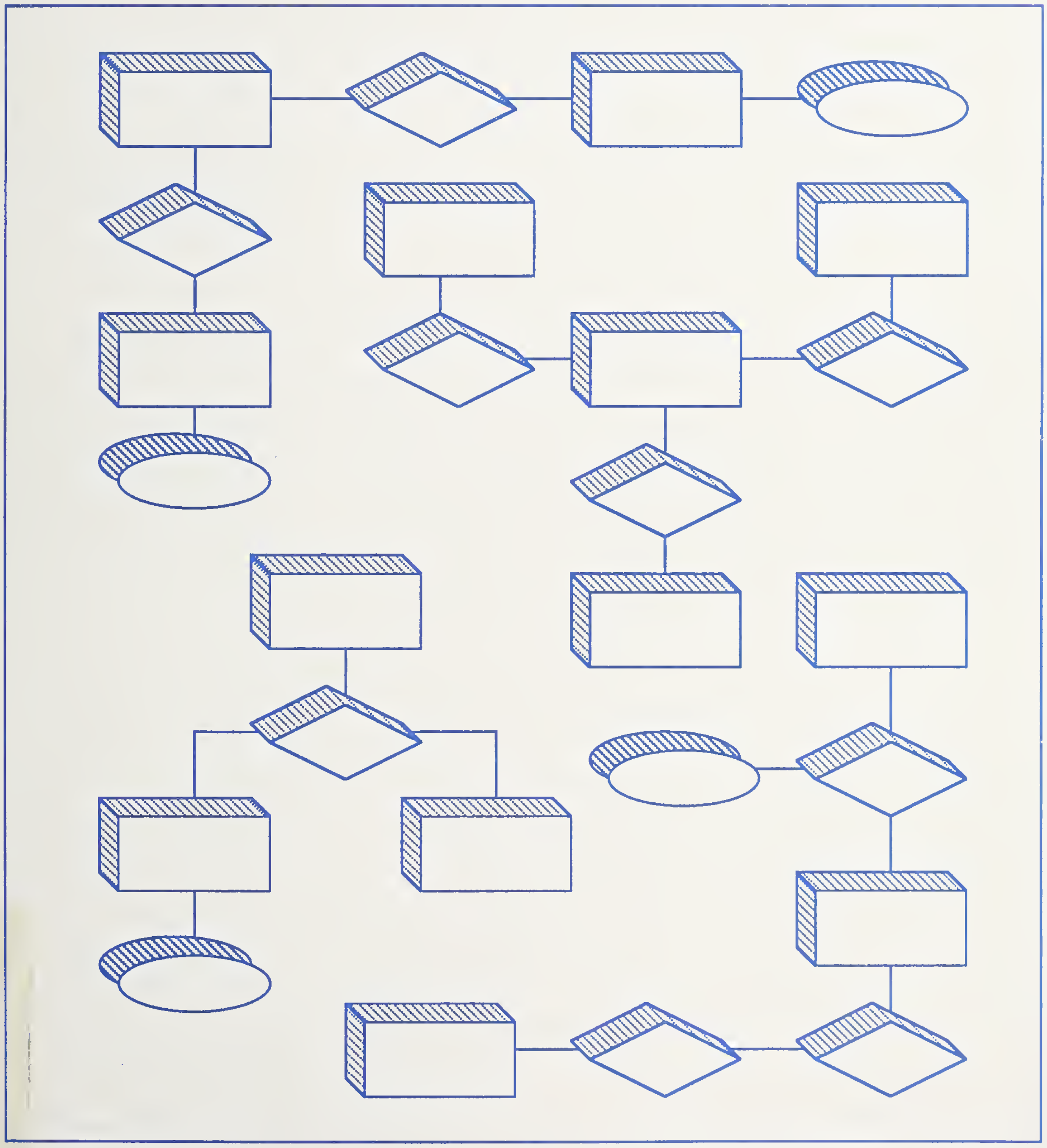
Margaret H. Law

Technology

U.S. DEPARTMENT OF COMMERCE

National Institute of Standards and Technology

NIST



The National Institute of Standards and Technology[1] was established by an act of Congress on March 3, 1901. The Institute's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Institute conducts research to assure international competitiveness and leadership of U.S. industry, science and technology. NIST work involves development and transfer of measurements, standards and related science and technology, in support of continually improving U.S. productivity, product quality and reliability, innovation and underlying science and engineering. The Institute's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the National Computer Systems Laboratory, and the Institute for Materials Science and Engineering.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; provides calibration services; and manages the National Standard Reference Data System. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Computing and Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[3]

The National Computer Systems Laboratory

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Laboratory consists of the following divisions:

- Information Systems Engineering
- Systems and Software Technology
- Computer Security
- Systems and Network Architecture
- Advanced Systems

The Institute for Materials Science and Engineering

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-cutting scientific themes such as nondestructive evaluation and phase diagram development; oversees Institute-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Institute consists of the following divisions:

- Ceramics
- Fracture and Deformation[3]
- Polymers
- Metallurgy
- Reactor Radiation

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.
[2]Some divisions within the center are located at Boulder, CO 80303.
[3] Located at Boulder, CO, with some elements at Gaithersburg, MD.

NISTC
QC100
.U57
no. 500-1
1989
c.2

NIST Special Publication 500-173

Guide to Data Administration

Bruce K. Rosen
Margaret H. Law

National Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

October 1989

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Raymond G. Kammer, Acting Director

NIST

Reports on Computer Systems Technology

Library of Congress Catalog Card Number: 89-600766
National Institute of Standards and Technology Special Publication 500-173
Natl. Inst. Stand. Technol. Spec. Publ. 500-173, 81 pages (Oct. 1989)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1989

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

ABSTRACT

Data Administration is the management of information describing the data, functions, operations, and structure of automatic data processing (ADP) systems and databases. Data Administration collects descriptive information about an organization's data, functions, and operations to support the development of the ADP systems, to provide full system documentation during systems operation and maintenance, and to support redesign efforts required as the needs of an organization evolve.

This guide provides a reference model for the various activities performed by Information Resource Management, Data Administration, Data Modeling Tools Administration, and Database Administration. The functions of Data Administration are discussed in detail. Data Administration is responsible for defining an information architecture, and for establishing policies for naming conventions, information modeling techniques and methodologies, data element specification, system information integration, and data protection. The importance of the Three Schema Architecture is described.

Computing tools useful for Data Administration, such as data dictionary systems and computer-aided software engineering (CASE) tools, are addressed. The Guide stresses the useful features of these computing tools. Use of the Information Resource Dictionary System (IRDS) standard, approved by the American National Standards Institute (ANSI) and a Federal Information Processing Standard (FIPS), is discussed.

KEY WORDS

Data administration, database administration, data management, data modeling, information resource dictionary system, IRDS, information resource management, IRM.

ACKNOWLEDGMENTS

We gratefully acknowledge the contributions of David Jefferson, Judith Newton, Alan Goldfine, and Frank Spielman, who were instrumental in the composition of this guide. Without their help, this guide would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This guide was developed to assist Data Administration professionals in applying current and future technology to manage and control the information explosion occurring within organizations today.

1.1 Introduction to Data Administration

Data Administration is the management of information describing the data, functions, operations, and structure of both manual and automatic data processing systems (ADP) and databases. The descriptive information about an organization's systems and data is used to support the development of ADP systems, to provide full system documentation for use during systems operation and maintenance, and to support redesign efforts required as the needs of an organization evolve.

The Data Administration function is performed throughout the information system lifecycle: (1) during the system development phases, (2) during the system operations and maintenance phase, and (3) during the system redesign phase.

Since an information system is often comprised of multiple subsystems, which may be undergoing work in different lifecycle phases, the Data Administration of an information system may be simultaneously involved in all phases of the system lifecycle.

1.2 Scope of the Guide

As a first step, this guide provides a reference model that discusses the role of Data Administration within the context of an organization. The information management reference model discusses the various responsibilities assigned to the areas of Information Resource Management, Data Administration, Data Modeling Tools Administration, and Database Administration. The guide then looks at the various external interfaces through which Data Administration interacts with other organizational units, and examines the activities that Data Administration must undertake to respond to those interfaces. The functions of Data Administration are then discussed.

Data Administration is responsible for defining the data component of the Information Resource Management architecture, which may be based on a three schema data architecture. The Three Schema Architecture includes a definition of: (1) the Conceptual Schema representing all the information resources of an organization, (2) the External Schema(s) representing each department's view of the information resources, and (3) the Internal Schema(s) representing the physical storage of information resources, perhaps in multiple databases. Many of the data structures, or schemas, that Data Administration develops will fit within the framework of such a Three Schema Architecture. Some other Data Administration functions include performing external activities, performing coordinating activities, and using computing tools to enhance data management performance.

Data Administration is responsible for establishing policies for naming conventions, data element specification, system information integration, and data protection. Along with the Software Engineering function, Data Administration is responsible for establishing policy for information modeling techniques and software development methodologies. These policies, and the use of computing tools to carry out these policies, are discussed.

Computing tools useful for Data Administration, such as data dictionary systems, also known as repositories, and computer-aided software engineering (CASE) tools, are addressed. The guide stresses the useful features of these classes of computing tools that can improve data management performance.

The Information Resource Dictionary System (IRDS) Standard is discussed in terms of its support for Data Administration. The IRDS standard has been approved by the American National Standards Institute (ANSI) and has been adopted as Federal Information Processing Standard (FIPS) 156. The relationship of the IRDS Standard to data dictionary systems and CASE tools, is explained. The role of Data Administration in supporting Configuration Management, particularly in supporting computer software and hardware Configuration Management, is addressed.

This document does not include a survey of the computing tools available in the general marketplace today. Instead, the guide discusses the types of functionality that are useful in computing tools applied to Data Administration. Where appropriate, this guide addresses software standards, reports, guidelines, and other publications that may be useful in the performance of Data Administration.

2. REFERENCE MODEL

A reference model is included to provide a logical framework for the information management structure within an organization.

2.1 Purpose

This reference model provides a simple framework on which to base a discussion of the various information management roles performed to support the information needs of an enterprise.

The information management reference model is divided into four parts: (1) Information Resource Management, (2) Data Administration, (3) Data Modeling Tools Administration, and (4) Database Administration. While these information management roles are referred to by this guide, it is recognized that organizations will have various reasons to order their information management structures differently, with different areas of emphasis. Although the Software Engineering function is not addressed in this guide, it is essential that Software Engineering and Data Administration coordinate efforts and work together during the system lifecycle.

2.2 Information Resource Management (IRM)

IRM is the management function responsible for corporate level control of all information resource aspects of an organization. A variety of responsibilities may be managed under the IRM umbrella. Some IRM functions may include responsibility for computer software purchases, computer software development and maintenance, computer hardware purchases and maintenance, personnel resource assignments, etc.

In making decisions concerning these different responsibilities, IRM must not only be concerned with high-level technical information resource issues, but also with a number of management issues involving the enterprise at large. For example, IRM must be concerned with implementing corporate goals, following corporate growth strategies, earning the expected return on investment, keeping expenditures within the limits of funding availability, training and retaining personnel, and keeping programs within projected limits for physical growth.

While high-level technical information resource issues must be involved in IRM decision-making, "real world" management decisions are often based on other concerns. Accordingly, higher level management decisions may sometimes be reached without much consideration for technical issues. Even issues such as personal influence and corporate "pecking order" can have a greater impact than technical issues on the decisions reached at this management level.

IRM must operate in the area between high-level management's perspective of the enterprise as a whole and the technical, operational perspective of managing the organization's information resources. IRM interprets high-level management's directives in terms of information resource management policy for Software Engineering and Data Administration. IRM determines what applications and data are needed to support the organization's requirements. Software Engineering and Data Administration then determine how those applications are to be developed, and how data is to be stored, accessed, and maintained. Data Administration and Software Engineering also provide some technical advice on managing information resources to IRM.

2.3 Data Administration

Data Administration is concerned with the definition, control, and effective use of all information resources within an organization. One of Data Administration's primary areas of responsibility is establishing and maintaining the flow of information into, within, out of, and among the information systems used by an organization.

Information collection, structure, storage, and access, even if not automated, are other areas of Data Administration responsibility. It is the responsibility of Data Administration to ensure that the data being collected and maintained within an organization is structured and stored in such a manner that it is accessible and understandable to all segments of the organization that have a legitimate interest in the information. To this end, with the cooperation of Software Engineering, Data Administration establishes and enforces organization-wide policies and standards involving data and system-related information.

Data Administration is responsible for ensuring that adequate records are kept to reflect accurately the "who, what, when, where, why, and how" of all data being collected, manipulated, or maintained within the organization. Finally, with the cooperation of Software Engineering, it is the responsibility of Data Administration to assign the available data system and storage resources to best serve the needs and priorities of the entire organization.

IRM is responsible for developing information resource policies used throughout the entire enterprise. Data Administration enforces those policies, and, in turn, also directs the functions of Database Administration and Data Modeling Tools Administration.

Data Administration directs the function of Database Administration by overseeing a number of Database Administrators, each of whom is responsible for one or more particular databases. A database may be a database application implemented on a database management system (DBMS), or it may be a collection of "flat files" used together and collectively referred to as a database.

In the performance of Data Administration, automated data modeling tools are often used. Data Administration directs the function of Data Modeling Tools Administration, by overseeing a number of Application Administrators and Dictionary Administrators. An Application Administrator is an individual who is responsible for maintaining a particular data modeling application on a CASE tool. Similarly, a Dictionary Administrator is responsible for maintaining a particular dictionary application on a data dictionary system or repository tool. If it is more appropriate to the organization, however, Data Administration may oversee a number of Tool Administrators, each of whom is responsible for one or more software tools.

2.4 Data Modeling Tools Administration

While Data Administration is responsible for interpreting and enforcing information resource policies for the enterprise, Data Modeling Tools Administration is responsible for managing and carrying out those policies for individual data dictionaries, repositories and other applications of modeling tools.

Data modeling tools are used to develop and describe databases, information systems, and information interchange within and among systems. Data Modeling Tools Administration directly supports the Data Administration process.

In organizations that have integrated data modeling applications, Data Modeling Tools Administration may be implemented on an application basis. In this case, Application Administrators and Dictionary Administrators should carry out data administration procedures and policies for each data modeling tool application.

In organizations that have multiple tools and a proliferation of data modeling applications, Data Modeling Tools Administration may be implemented on the basis of particular automated tools. In this case, individual Tool Administrators should carry out data administration policies for applications running on each data modeling tool. Each data modeling tool can be used to support multiple data modeling applications.

2.5 Database Administration

While Data Administration is responsible for interpreting and enforcing certain information resource policies for the enterprise as a whole, Database Administration is responsible for managing and carrying out those policies for individual subject databases.

Database Administration is concerned with the efficient use of the resources that hold and make available the data resources of particular databases within an organization. This is the area responsible for controlling and maintaining the flow of data into, within, out of, and among each individual database.

Database Administrators physically control how, where, and in what manner, data is stored and maintained within each database. Each Database Administrator ensures that his or her database is operating smoothly and optimally, with a good physical database design that reflects the current state and usage of the database.

At the same time, Database Administrators provide information to Data Administration concerning organizational use of data within the subject database. In touch with database users, Database Administrators are responsible for recognizing when and how users' needs change, and if these changes require modification of the subject database. When users' needs evolve significantly, a Database Administrator reports these changing requirements to Data Administration. If users' needs change significantly, Data Administration may decide that one or more existing databases require redesign, or a new database is required.

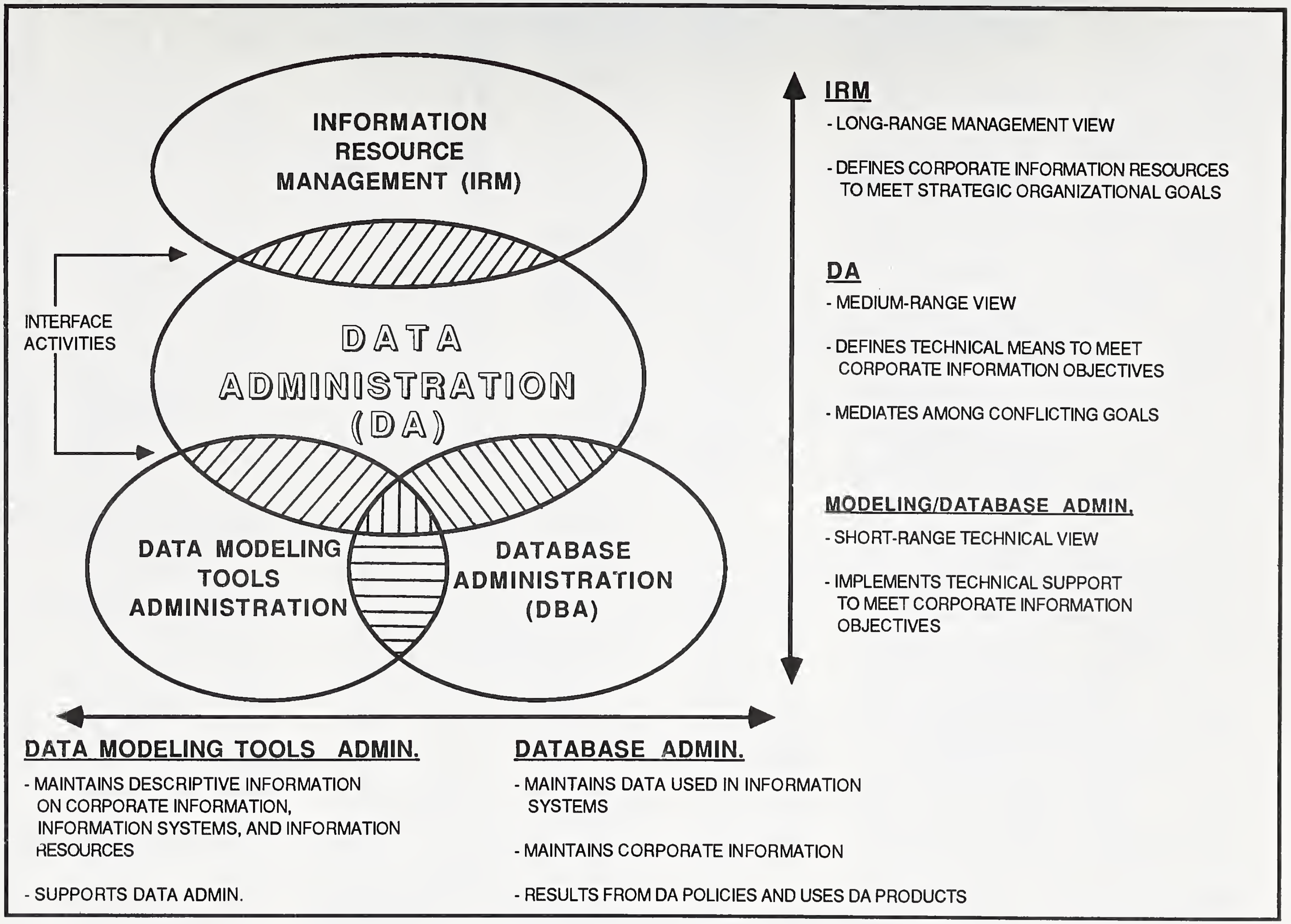
2.6 Data Management Integration

These four organizational areas combine to create an interconnected hierarchical structure. Figure 1, Information Management Integration, illustrates how these four areas are structured and interconnected with shared interface activities.

Figure 1 shows that IRM is primarily concerned with the long-range management view of the enterprise. To this end, IRM defines the necessary corporate information resources to meet the enterprise's strategic organizational goals. While IRM should be sensitive to technical issues brought up by Data Administration, IRM must adjust information resource management policy to follow the directives of corporate management.

Data Administration, on the other hand, is shown with a mid-range view of the enterprise, intermediary between the long-range management view of IRM and the sometimes short-range technical view of information management implementation. Following IRM directives, Data Administration defines the technical means necessary to meet corporate information objectives, brings technical considerations to the attention of IRM, and mediates between potentially conflicting technical and strategic goals. To maximize the benefit of data resources, Data Administration defines organization-wide policies for Database Administration and Data Modeling Tools Administration.

Data Modeling Tools Administration directly supports the Data Administration process, and maintains descriptive


INFORMATION RESOURCE MANAGEMENT (IRM)
INTERFACE ACTIVITIES
DATA ADMINISTRATION (DA)
DATA MODELING TOOLS ADMINISTRATION
DATABASE ADMINISTRATION (DBA)

IRM

- LONG-RANGE MANAGEMENT VIEW
- DEFINES CORPORATE INFORMATION RESOURCES TO MEET STRATEGIC ORGANIZATIONAL GOALS

DA

- MEDIUM-RANGE VIEW
- DEFINES TECHNICAL MEANS TO MEET CORPORATE INFORMATION OBJECTIVES
- MEDIATES AMONG CONFLICTING GOALS

MODELING/DATABASE ADMIN.

- SHORT-RANGE TECHNICAL VIEW
- IMPLEMENTS TECHNICAL SUPPORT TO MEET CORPORATE INFORMATION OBJECTIVES

DATA MODELING TOOLS ADMIN.

- MAINTAINS DESCRIPTIVE INFORMATION ON CORPORATE INFORMATION, INFORMATION SYSTEMS, AND INFORMATION RESOURCES
- SUPPORTS DATA ADMIN.

DATABASE ADMIN.

- MAINTAINS DATA USED IN INFORMATION SYSTEMS
- MAINTAINS CORPORATE INFORMATION
- RESULTS FROM DA POLICIES AND USES DA PRODUCTS

FIGURE 1. INFORMATION MANAGEMENT INTEGRATION

information on an enterprise's information, databases, information systems, and other information resources. Data modeling tools are used in the development and maintenance of databases and information systems. Data Modeling Tools Administration is concerned with ensuring that Data Administration users of data modeling tools are effectively and efficiently supported in their data modeling applications.

Database Administration maintains corporate information and data used in information systems. Database Administration is primarily concerned with making certain that database users are served efficiently and effectively by the various components that make up a database, even if the Database Administration area does not itself directly provide the service. Database Administration follows the policies of Data Administration, bringing problems and necessary changes to the attention of the Data Administrator.

2.7 Organizational Interfaces

Figure 2, Organizational Interfaces, shows how these four aspects of the information management model interface with each other and with various other components of the organization.

At the top of figure 2, the high-level management of the organization sets the strategic goals for the enterprise and develops the corporate plans to achieve those strategic goals. From IRM, high-level management receives requirements for resource commitments and requests for policy guidance on corporate strategic goals.

IRM receives resource requirements from Data Administration, commits enterprise resources to corporate endeavors, and defines organizational policy for Data Administration. In turn, Data Administration provides organizational information resource requirements and policy to Database Administration and to Data Modeling Tools Administration. Data Administration is supported by Data Modeling Tools Administration and the tool set that it operates.

Since Data Administration enforces corporate-wide policy for information management, the various databases within the organization can be combined to form a corporate-wide integrated data resource. If implemented in the data modeling tool set, a data directory can be used to provide access to particular information within the integrated data resource. Similarly an information system directory can be implemented in the data modeling tool set, to provide access to particular systems and subsystems available within the organization.

Data Administration provides descriptive modeling information that defines both information systems and the

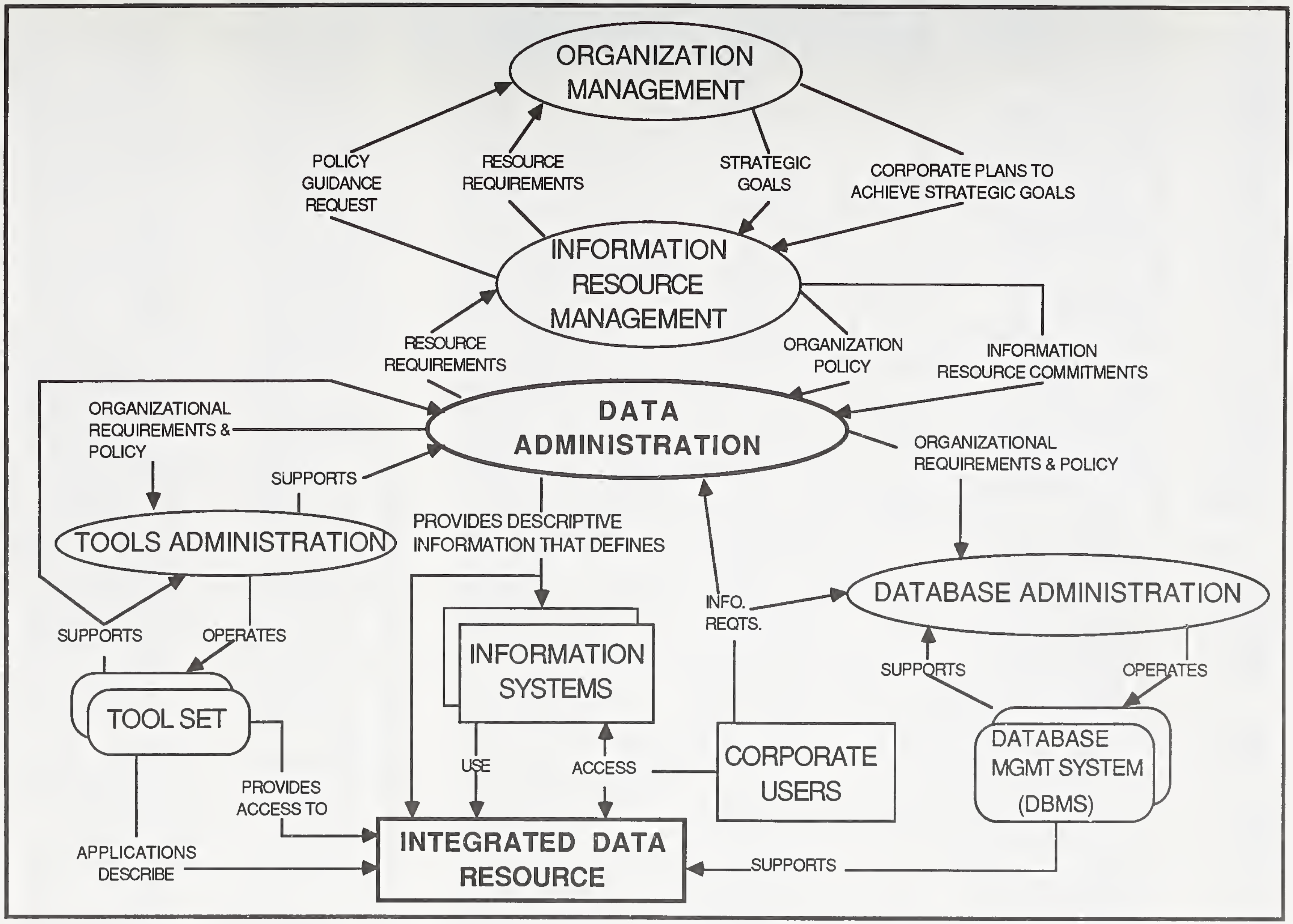


FIGURE 2. ORGANIZATIONAL INTERFACES

integrated data resource. Instead of using fragmented and dispersed data resources, information systems utilize the corporate integrated data resource.

Data Administration interprets and enforces the policy that determines which components of the organization may access which aspects of the organization's data assets, and when new database assets will be developed. Thus Data Administration's interface with the other portions of the organization will be at the higher policy-setting levels of the organization. Information users come to Data Administration to request access to different areas of the organization's information assets or to request that new assets be developed.

Database Administration operates database management systems (DBMSs) and other software supporting databases that, in turn, support the corporate integrated data resource. When corporate system users define their information requirements with the assistance of Data Administration, their local information requirements are handled by Database Administration, and their corporate information requirements are handled by Data Administration. Corporate system users access corporate information systems as part of the integrated data resource, which may be supported by one or more DBMSs.

In a sense, Database Administration serves as a valve that controls and directs the information flow between the actual physical databases of the organization, and the areas of the organization requesting access to that information. Also Database Administration is responsible for implementing any new database requirements defined by the organization. This situation results in Database Administration having a very direct interface with much of the working level of the organization in order to resolve the daily problems of accessing information.

3. DATA ADMINISTRATION FUNCTIONS

As mentioned previously, Data Administration interfaces with IRM, Data Modeling Tools Administration, Database Administration, and the corporate users of information resources. Each of these different interfaces exist because organizational requirements result in Data Administration sharing activities with these other components of the organization.

For example, a new requirement to verify the value of a portion of the enterprise's information resources might cause Data Administration to initiate "redundancy checking" as part of its interface with Database Administration. If it were found that one type of stored information could be calculated or derived from some other data stored within the corporate resource, then the former type of information would be considered redundant and eliminated from the appropriate database. The method of calculating or deriving that type of information would then be saved and utilized in the future.

3.1 IRM Architecture Definition

To succeed in its mission to manage information through the system lifecycle, Data Administration must first develop the data component of an IRM architecture. This IRM architecture should provide a framework for defining, describing, and analyzing the information resources of the organization, their relationships with each other, and the ways in which they support the organization. A major component of the Data Administration aspect of an IRM architecture is the ANSI/SPARC Three Schema Architecture.

3.1.1 ANSI/SPARC Three Schema Architecture

In the middle 1970's, the Standards Planning and Requirements Committee of the American National Standards Committee on Computers and Information Processing, abbreviated as ANSI/SPARC, developed what has become known as the "Three Schema Architecture." This architecture is a principle of database management by which Data Administration can logically structure the documentation of the enterprise's information resources, their relationships to each other, and to the organization.

Figure 3 depicts in the Three Schema Architecture in simple terms. The box labeled "Conceptual Schema" represents Data Administration's unified view of the total information resources of the organization. The Conceptual Schema, or logical view, is the result of integrating all the information models developed to describe the various aspects of the corporate information resource.

Those boxes labeled "External Schema" represent the ways in which different groups of users view the portions of the corporate information resource relevant to them. External Schemas are also referred to as "user views." When we think about the data used by a particular system, or the information resulting from a particular system, we are thinking in terms of an External Schema or user view.

Finally, the boxes labeled "Internal Schema" represent the actual physical storage of information resources, which may be stored on multiple, dispersed databases. These multiple databases may even be using totally different storage methods.

The Conceptual Schema has an organization-wide scope, and is quite stable with respect to changes in IRM technology or

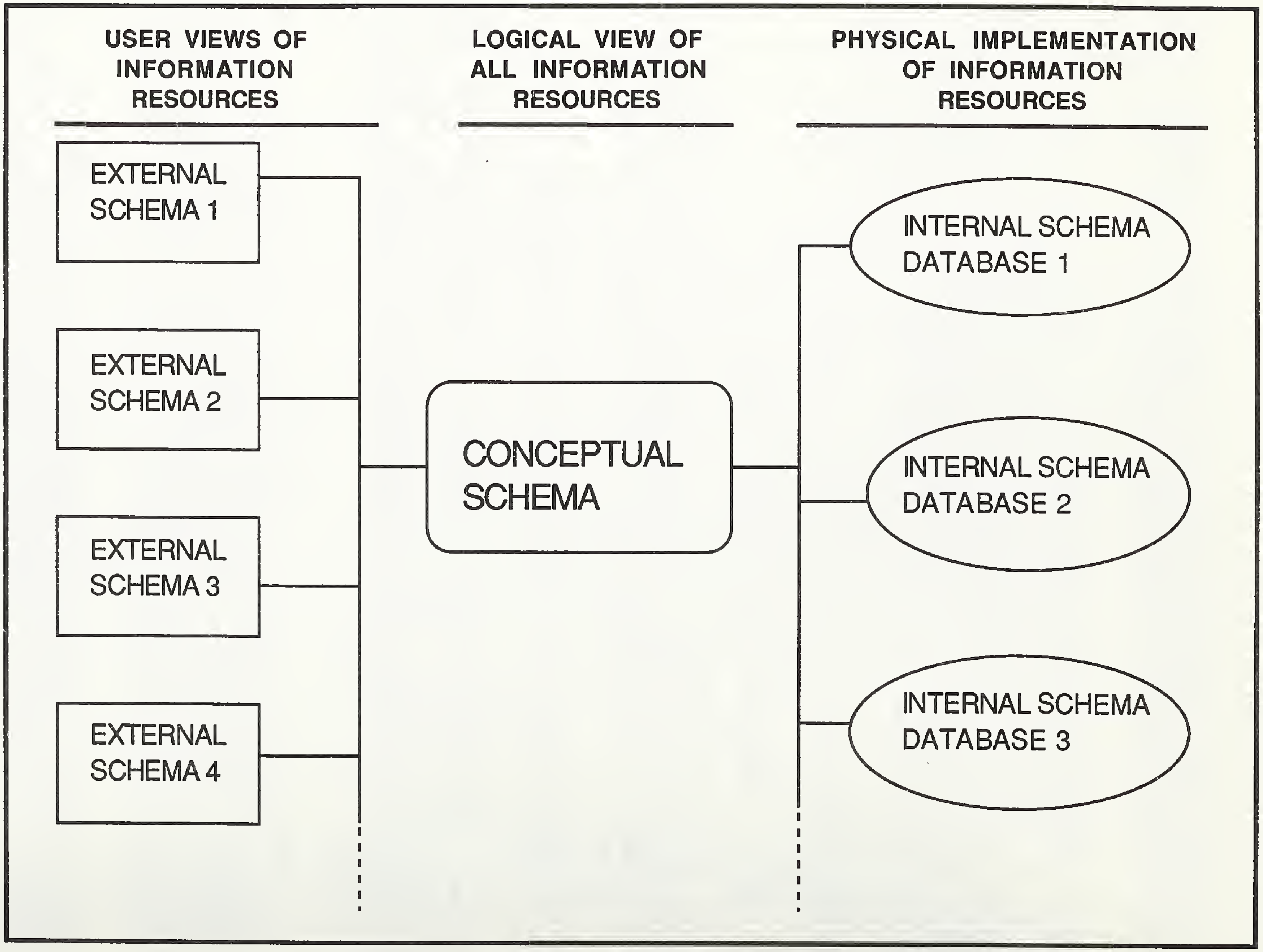

USER VIEWS OF INFORMATION RESOURCES
LOGICAL VIEW OF ALL INFORMATION RESOURCES
PHYSICAL IMPLEMENTATION OF INFORMATION RESOURCES
EXTERNAL SCHEMA 1
EXTERNAL SCHEMA 2
EXTERNAL SCHEMA 3
EXTERNAL SCHEMA 4
CONCEPTUAL SCHEMA
INTERNAL SCHEMA DATABASE 1
INTERNAL SCHEMA DATABASE 2
INTERNAL SCHEMA DATABASE 3

FIGURE 3. THREE SCHEMA ARCHITECTURE

specific applications. The External Schemas have a narrow scope, change as applications change, but are stable with respect to IRM technology. Finally, the Internal Schemas change as IRM software and hardware changes, but remain relatively stable with respect to changes in applications. Internal Schemas change in response to changing database access workloads, since alterations in workload usually require fine-tuning the physical database design to maximize performance. For a more complete discussion of the Three Schema Architecture refer to [JARD77].

3.1.2 Example Three Schema Application

As an example of a three schema architecture, consider the following scenario. An organization has data stored on its computers about its own personnel. The following are the field names for this personnel data:

- Employee Name
- Employee Home Address
- Employee Date Of Birth
- Employee Gross Salary
- Employee Taxes Withheld To-Date
- Employee Social Security Number
- Employee Spouse Name
- Department Number
- Employee Home Phone Number
- Employee Office Phone Number

Notice that "employee" is the first term in each field name that refers to employee information, except for Department Number. As will be discussed later under Naming Policy, "employee" is used here as a "prime word." Employee Name is used, instead of another term such as "Full Name," to differentiate Employee Name from other names such as Customer Name, Project Name, Product Name, etc.

Department Number does not have the term Employee at the beginning of the field name since Department Number refers particularly to departments, not just to employees. While an employee is associated with a particular Department Number, many other entities may be similarly associated. For example, one or more Department Numbers may be associated with a Project.

Taken in the order presented above, these field name entities could be considered to be the personnel portion of the Conceptual Schema maintained by Data Administration. Even though these data fields are presented together in the conceptual schema, the actual physical instances of the data fields could be scattered throughout several physically separated databases, with perhaps the individual's Social Security Number serving as a "key" to link together the data describing a particular employee.

Thus, in the Internal Schema representation of the organization's data resources, the storage of these field names may be dispersed. Employee Home Address and Employee Home Phone Number might be part of a database of sensitive personal information, while Employee Taxes Withheld To-Date might be stored with payroll information, and Employee Office Phone Number and Department Number could be part of an "Employee Directory" database.

Further, the diverse systems storing this information could be functioning under completely different modes of operation. For example, the sensitive personal information might be stored in a database running on a "relational" DBMS, while payroll information might be stored in a "flat file", and the data for the "Employee Directory" might be stored in a database running on a "hierarchical" or "network" DBMS.

The Payroll Department might regularly produce a report from what they believe is a single database containing the following information:

- Employee Name
- Department Number
- Employee Social Security Number
- Employee Gross Salary
- Employee Taxes Withheld To-Date

The Payroll Department has what is referred to as an External Schema, or user view of the information assets of the organization. The Payroll Department only has to concern itself with the information needed to do its job, and not with all of the other information contained in the organization's information assets. This focus on the user view simplifies the programs and queries developed by Payroll and other departments.

The compartmentalization of information assets according to External Schema or user view also provides a basis for establishing measures to protect data integrity and security. Permission to read particular types of data is granted on the basis of a user's "need to know," which is determined partly by the user's External Schema view. Permission to change particular types of data, however, is determined on the basis of the user's authority to control certain data types, which may be established as a subset of a user's External Schema view.

3.1.3 Advantages of Three Schema Architecture

Figure 4, Different Views of Information Resources, illustrates another aspect of the ANSI/SPARC Three Schema Architecture. This figure is intended to emphasize that the primary purpose behind this model is to provide a methodology through which Data Administration can maximize the sharing of information resources throughout the enterprise.

By providing a relatively stable Conceptual Schema, this model provides Data Administration with a stable platform upon which to develop multiple External Schemas. Multiple External Schemas are needed to provide multiple user groups with views of their information assets. Further, these individual user views should be presented in a manner that can be understood by different users with different functional orientations. Similarly, the stable platform of the Conceptual Schema also allows Database Administration to develop the Internal Schemas required to take advantage of the variety of information storage technologies available.

It is this control architecture, represented by the Conceptual Schema, that provides the enterprise with the increased ability to share its information assets.

One of the most common problems within any organization is the fact that it is difficult for individuals to determine if their portion of the organization could utilize an already existing information asset. This level of difficulty is increased when the description of that information asset is oriented towards some other functional area of the organization. If these users try to examine descriptions of assets based on the types of storage technology under which the assets are maintained, the problem is magnified still further.

The organizationally neutral descriptions of information assets, presented in the control architecture, provide the needed common ground or common reference. With this area of common reference, individuals within the organization can review the organization's information assets to look for assets that their portion of the organization could share and utilize.

Returning to the simple example presented previously, the actual system that would generate the needed Payroll Department report might query, or generate reports from, one or more DBMS's to get the needed information. The query or report request would interact with each database's Internal Schema. The Internal Schema for the information contained in each database is represented in the physical storage structures of each DBMS database application.

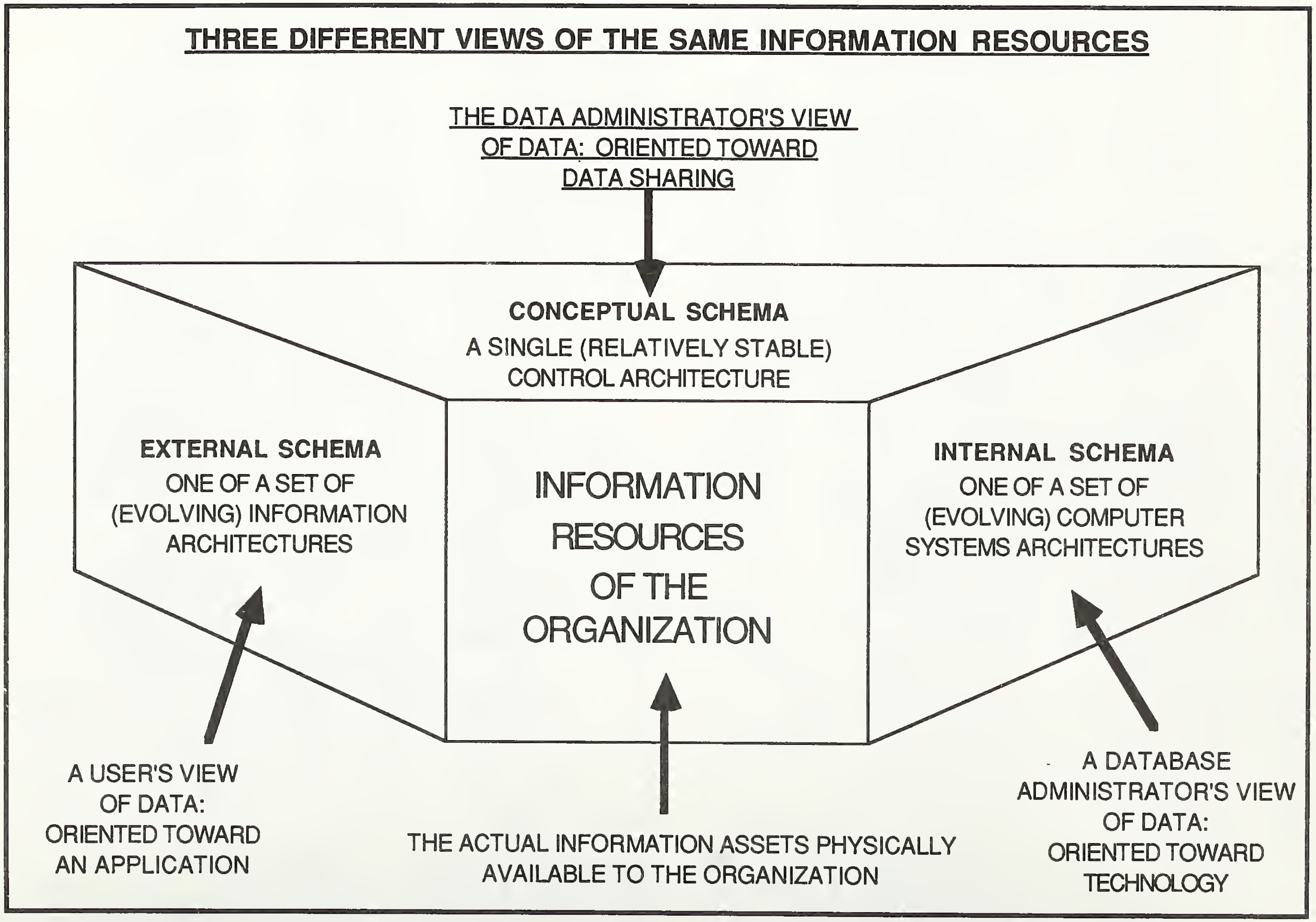

THREE DIFFERENT VIEWS OF THE SAME INFORMATION RESOURCES
THE DATA ADMINISTRATOR'S VIEW OF DATA: ORIENTED TOWARD DATA SHARING
CONCEPTUAL SCHEMA
A SINGLE (RELATIVELY STABLE) CONTROL ARCHITECTURE
EXTERNAL SCHEMA
ONE OF A SET OF (EVOLVING) INFORMATION ARCHITECTURES
INFORMATION RESOURCES OF THE ORGANIZATION
INTERNAL SCHEMA
ONE OF A SET OF (EVOLVING) COMPUTER SYSTEMS ARCHITECTURES
A USER'S VIEW OF DATA: ORIENTED TOWARD AN APPLICATION
THE ACTUAL INFORMATION ASSETS PHYSICALLY AVAILABLE TO THE ORGANIZATION
A DATABASE ADMINISTRATOR'S VIEW OF DATA: ORIENTED TOWARD TECHNOLOGY

FIGURE 4. DIFFERENT VIEWS OF INFORMATION RESOURCES

The requested information would then be presented in a report form that represents the Payroll Department's External Schema view. At no time is the query or report generation program directly concerned with producing any information, or having any input, presented in the form of the enterprise's Conceptual Schema or in the form of a database's Internal Schema.

It is the existence of the Conceptual Schema, however, that allowed the Payroll Department to share previously existing information in the production of the needed report. The Conceptual Schema is incorporated into the design of the database running on a DBMS, to map the multiple Internal Schemas to the multiple External Schema views.

3.2 Data Administration External Activities

Figure 5, Data Administration Functions, represents some of the external activities, or interface activities, that occur between Data Administration and the areas of IRM, Database Administration, and other users. These interface activities are indicated by the arrows arrayed around the box representing the Data Administration function. Each set of arrows between Data Administration and one of the other areas shown, such as IRM, represents a subset of the entire set of interface activities between the two functions.

While each "named activity" may appear within more than one activity grouping, the requirements behind the "named activity" could be the same, or different, depending on the organizational requirements generating the activity. Accordingly, the actual work being performed under the "named activity," that appears in more than one activity group, might be quite different for each of the different activity groupings.

3.3 Data Administration Coordinating Activities

The generic Data Administration functions listed inside the box in figure 5 are not intended to be a total listing of all needed Data Administration functions. These are simply an initial set of functions for which any Data Administration area should develop expertise to fulfill organizational needs.

The levels of expertise and the amount of effort expended to perform these various Data Administration functions within any particular organization will vary, depending on many factors. Some factors that may influence the degree of difficulty in performing Data Administration functions include the size of the entire organization, the direction of corporate strategic decisions, levels of automated support, legal requirements, etc.

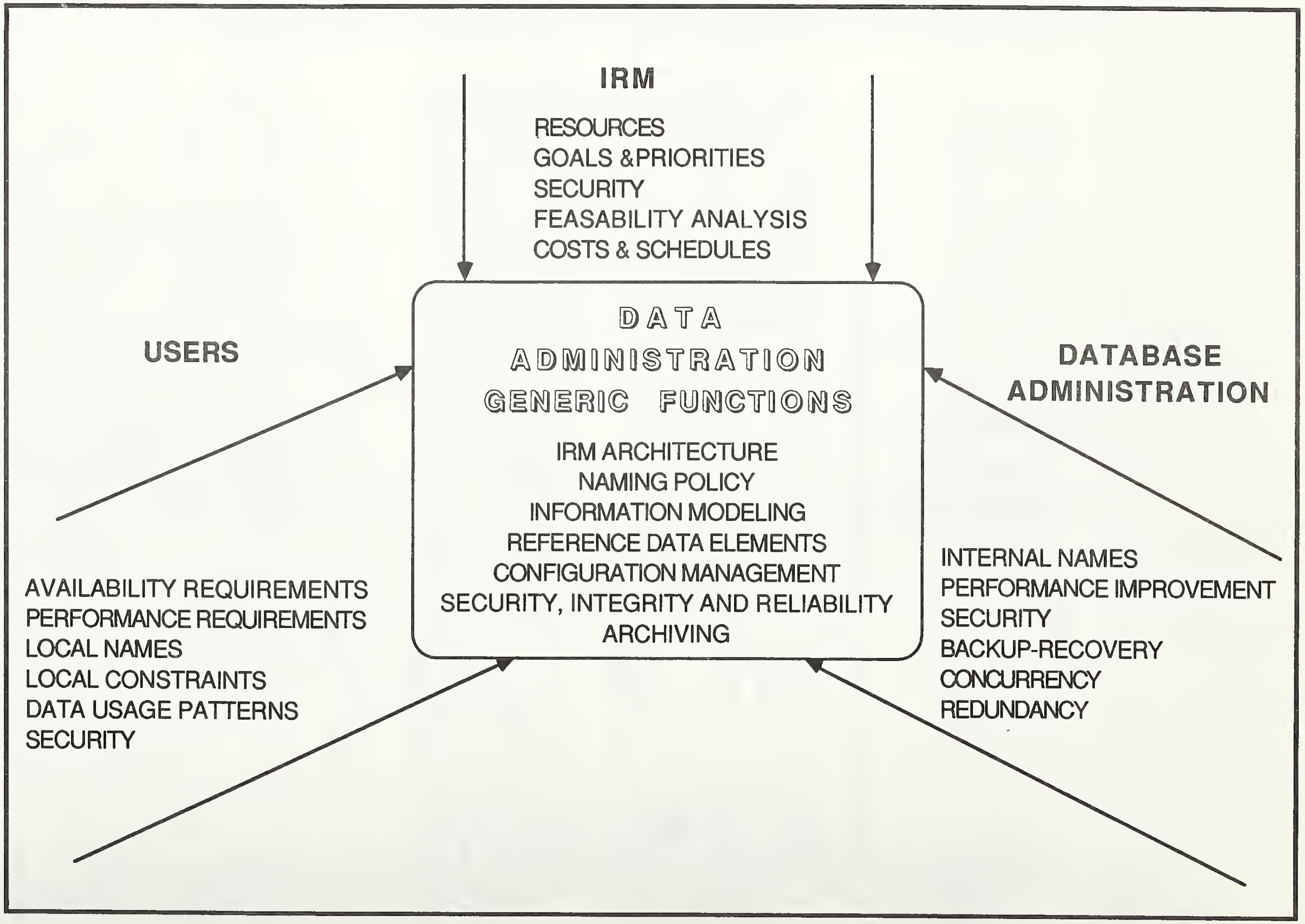


FIGURE 5. DATA ADMINISTRATION FUNCTIONS

3.3.1 Mapping Data Administration Functions to Activities

To satisfy the many requirements behind the various activities performed by Data Administration, these activities and their requirements must be organized and coordinated. The approach shown in figure 5 is to establish and define a set of generic Data Administration functions that can then be "mapped" to the appropriate organizational activity and its requirements.

As an example of this mapping effort, consider the "security" activity. This particular activity is shown under all three of the organizational area activity groupings. Yet the requirements for each of these three different "security" activities are quite different.

From the user's view, security requirements could be based on the user's concerns to protect against the change or loss of valuable working data, or the need to satisfy concerns of an outside party for whom the user is performing a service.

From the IRM perspective, security requirements could be based on the need to satisfy a number of legal requirements for protection of information, as well as the need to limit the expenditures on special security equipment.

From the Database Administration perspective, information security actions might be based on the requirement to avoid concentration of sensitive information on any one physical piece of storage media. This requirement would restrict the loss or compromise of information resulting from one action or error.

These differences in the requirements behind identically named activities could result in the mapping of these "security" activities to entirely different generic Data Administration functions. For example, while the user security requirement might map directly to the Data Administration security function, the Database Administration security requirement might map instead to the IRM architecture function.

3.3.2 Mapping Data Administration Activities to Functions

Further, to satisfy the requirements behind any one activity, it may be necessary to map that activity to more than one generic Data Administration function. Again as an example take the IRM "named activity" of security. While this activity would be mapped to the generic Data Administration function of security policy, it might also be mapped to other areas such as: "global names," which may be used to identify security related data elements, or "integrity and reliability policy," which may be affected by security directives issued by IRM.

As this example shows, even when a generic Data Administration function exists whose name is similar to a "named activity," the needed mapping may not simply be a one-to-one relationship, but may, in fact, be a one-to-many relationship.

3.4 Tool-Oriented Data Administration Functions

The following is a sampling of some of the generic Data Administration functions for which various tools might provide assistance:

- Establishing and enforcing a naming policy
- Information modeling
- Developing the model of an IRM architecture
- Specifying reference data elements
- Performing configuration management
- Establishing and enforcing policies for security, integrity, and reliability
- Establishing and enforcing policies for archiving information resources.

This list is not intended to be an exhaustive list of all generic Data Administration functions. Further, the content of any such list would change depending on such factors as organizational structure or size, type of organization, resources being automated, etc. Also, the items appearing on this list are in no way mutually exclusive. Every item on this list could influence, or have an impact on, every other item. For example, various decisions made concerning configuration management could easily affect naming policy or policy concerning reference data element specification.

Specific Data Administration functions are addressed in the following chapters. Since the performance of Data Administration depends on support from automated tools, the functionality of tools to support Data Administration is also addressed.

4. NAMING POLICY

To manage data, it is imperative that an organization-wide policy be established to provide unique entity names that have high information content within a limited number of characters. This is one of the generic Data Administration functions that can be enabled and enforced through use of automated Data Administration tools.

4.1 Establishing Naming Policy

Establishing a Naming Policy does not in itself require a specific type of software tool. Instead, what is needed is guidance on how to establish this policy for any organization. Guidance on data entity naming conventions is provided in [NEWT87], which describes methods for structuring the format and content of data entity names. This data entity naming methodology is intended to maximize the opportunities to analyze and share the data resources of an organization.

Since every organization has different naming requirements, [NEWT87] does not attempt to provide any one specific naming policy to be used by every organization. Instead, it provides the needed guidance that an organization can follow to establish a naming policy that is appropriate to each organization's naming requirements.

4.2 Naming Convention Development

At first glance, data entity names may seem no different from natural language nouns. But they differ from nouns in the same way programming languages differ from natural languages: by the constraints imposed upon them by hardware, software, and human users, and by the possibility for the expression of the organization of the data itself.

Data entity names can reflect the organization of the data both logically, through <u>prime words</u>, and associatively, through <u>class words</u>. Prime words represent the logical groupings of data, such as all information which describes the concept **employee**; class words describe the basic nature of a class of data, such as **name**, **code**, or **date**. Data elements, one type of entity, may need a set of class words to fully describe all elements, while other entities such as **file** or **record** may need only one. <u>Modifiers</u>, which establish uniqueness of the data entity name, are the third name component.

While there may be many rules to be established for a set of naming conventions, there are a few guiding principles to follow while writing those rules:

Clarity - names are as clear as possible to a casual user.

Brevity within uniqueness - names are short while still maintaining uniqueness within the database.

Conformance to rules of syntax - each name is in the proper format. If there are too many names which cannot be made to fit the naming conventions, the rules may be too rigorous.

Context-freedom - each name is free of the physical context in which the data entity is implemented.

The current Information Resource Dictionary System (IRDS) standard, which is discussed in detail in a later section, provides an initial basis on which Data Administration can apply its own naming conventions. This is done by defining a framework for establishing the structure of names for each entity and their relationships to each other (i.e., the <u>metanaming structure</u>). There are three types of names for each entity: <u>access name</u>, <u>descriptive name</u>, and <u>alternate name</u>.

The access and descriptive names are functionally identical, but by providing two names, the IRDS allows them to share the burdens of the guiding principles of clarity and brevity. The access name may be terse, with abbreviations and acronyms but no connectors allowed (for example, EMPLOYEE-NAME), while the descriptive name allows for a longer and more discursive style (NAME OF EMPLOYEE). A user familiar with the database may want to use the access name for retrievals, while a more casual user would prefer the descriptive name. The alternate name may encompass any number of contingencies, such as physical name(s), report header name, and form input name. The majority of this discussion about names is concerned with access name grammar and usage.

The <u>content</u> component of naming grammar has been discussed above; the other component is <u>format</u>. Establishing format rules completes the process by which naming consistency is achieved. For instance, if the prime word is always the first word in the name and the class word last, there is no ambiguity in their identification. Searching by logical group (prime word) or basic nature (class word) is greatly simplified.

Use of naming conventions assists the Data Administrator in the analysis of data, for instance, by facilitating identification of coupled data elements and their decomposition into atomic data elements; and restructuring data names in which data is mixed in with metadata. Metadata is the specific type of data used to provide a description of other information resources (i.e., data about data).

A hierarchy of data elements can be developed based on class words. A "kernel" of class words can be used to form a set of standard or <u>generic</u> elements. These generic elements consist of a class word and modifier combination. Full data elements, called <u>application</u> elements, can then be formed with the addition of a prime word and any extra modifiers as needed.

As an example see figure 6, Entity Name Development. An application element EMPLOYEE-BIRTH-STATE-NAME is formed of the kernel class word NAME, which is contained in the generic element STATE-NAME; the prime word EMPLOYEE; and the modifier BIRTH. The alternate name for this element could be the abbreviation EMP-BTH-ST-NAM. Descriptive names are derived from access names by casting the access names into natural language grammar and adding connectors as needed. It is important to retain the prime and class words. For instance, EMPLOYEE-BIRTH-STATE-NAME becomes NAME OF BIRTH STATE OF EMPLOYEE.

Like most design activities, the effort expended in advance of the application of data entity naming conventions will pay off over the life of the enterprise.

4.3 Automated Support for Naming Conventions

Only when a Naming Policy has been established can a tool be utilized as part of the generic Data Administration function determining Naming Policy. While the current IRDS standard does supply a location in which to record names built in accordance with the organization's naming conventions, a naming conventions tool must supply additional functionality. Some of the requirements of this tool would be:

- Synonym identification by a list of all similar names or a permuted word list
- Identification of nonstandard names
- Generation of standard names by application of the rules specified by Data Administration
- Application of abbreviation rules.

A description of an organization's information resources should be defined in one or more Information Resource Dictionary (IRD) applications residing on an IRDS. This is the information resource where the names of an organization's information assets should reside. Any tool developed to support naming policy must be integrated with the organization's IRD, which describes these names. At this time, work has begun on an IRDS optional module to support naming convention verification.

5. INFORMATION MODELING

Once an IRM Architecture has been devised, and a naming policy has been established for the information to be managed, Data Administration can begin the information modeling process.

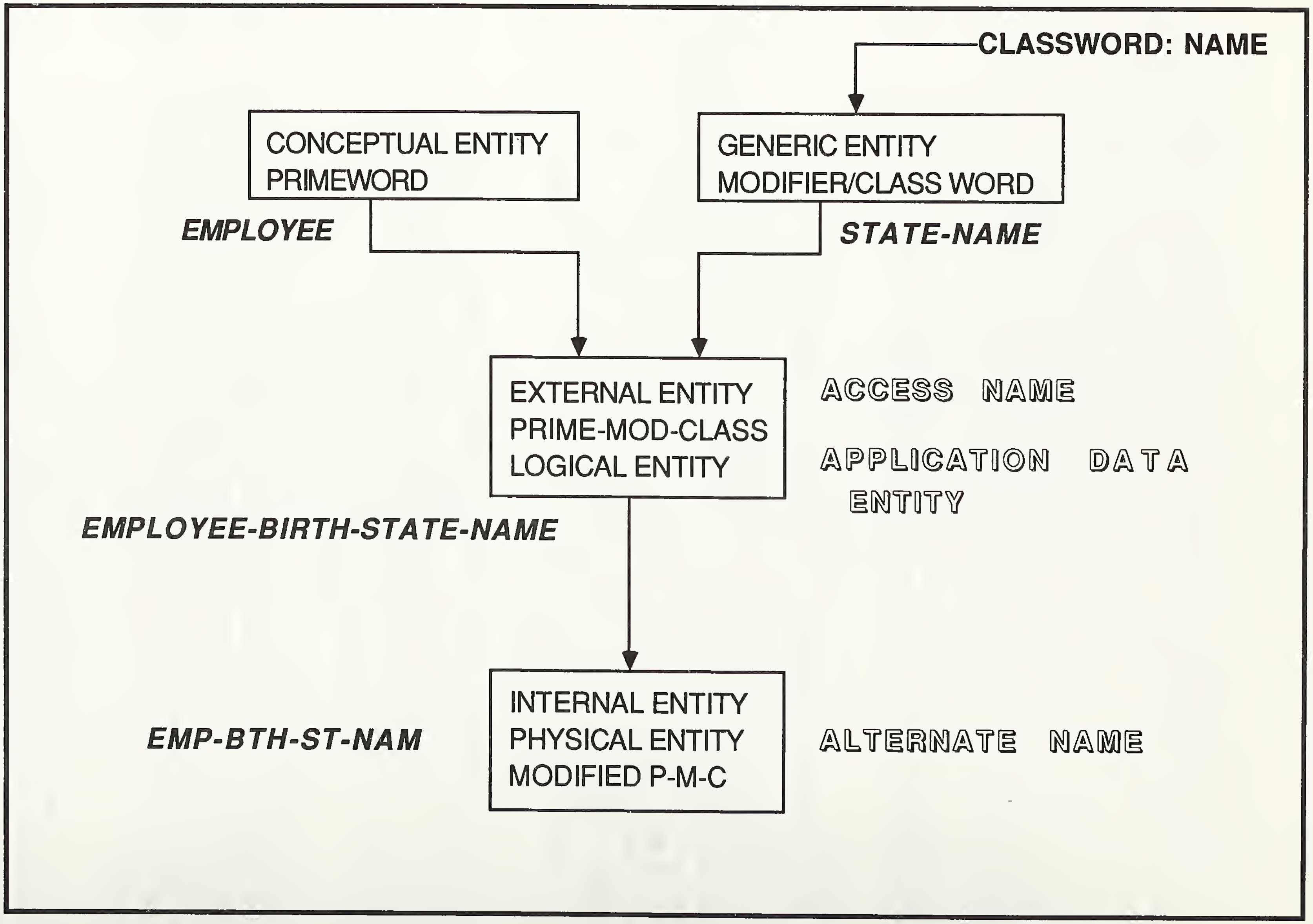


FIGURE 6. ENTITY NAME DEVELOPMENT

A model is the representation or simulation of an object or an activity; it is "a description or analogy used to help visualize something that cannot be directly observed" [WEBS83]. Information modeling is the representation of the procedures and results of systems analysis performed during the system lifecycle. An information model records the structure of an information system from phase to phase of its development.

5.1 Purposes of Information Modeling

As a Data Administration activity, information modeling is performed to capture and represent the information used in the development, operation, and maintenance of an information system. Information modeling provides a means of illustrating the structural concepts of system information. The Three Schema Architecture, described previously, is an example of an information model developed to illustrate three different types of perspective on the same information resource.

For Data Administration, the information generated through the process of information modeling should define and describe the information resources of the enterprise. These information definitions and descriptions are called "metadata," meaning "data about data." The metadata resulting from information modeling is stored using automated tools for further analysis, description, and access.

A variety of modeling techniques can be used in the representation of information modeling. Currently, some of the most popular modeling techniques are: (1) Entity-Relationship-Attribute models, which are used for database design as well as for general modeling; (2) Structured Analysis and Design models, which are used to support the full lifecycle; and (3) Object-Oriented models, which have become popular in the last few years and are used to support the full lifecycle.

Actual uses of these modeling methodologies and techniques, while recognizable as to their origin, will vary somewhat to suit different types of applications. Each of these models encompasses a number of variations in concept, approach, and graphical notation.

5.2 Modeling Through the System Lifecycle

There are many kinds of information models. Different information models are appropriate for different phases of the system lifecycle. Information modeling is a collection of techniques used to capture the descriptive system information generated over the course of the lifecycle.

Information models may include descriptions of:

- Information resources available or required to fulfill particular functions performed within an organization
- Data manipulation processes used to achieve these particular functions
- Relationships between the functions performed within an organization and the organization's products
- Types of information exchanged among different components of an organization.

Information modeling techniques are intended to span the system lifecycle. Information modeling should be performed during the strategic systems planning and requirements analysis phases, during the logical and physical database design phases, and during the program design and program implementation phases.

If information modeling was not performed or completed during system development, it can be performed retroactively during systems operations and maintenance, which is sometimes referred to as redesign. At every phase of the lifecycle, a complete information model of a system should be developed and maintained to provide a full description of the system's structure to be used during system maintenance and design upgrades.

Because Data Administration is performed throughout the lifecycle, Data Administrators should be familiar with the variety of modeling techniques and methodologies used throughout the lifecycle. General knowledge of different information modeling approaches will help Data Administrators to understand, plan for, and support the types of information resource data to be described and stored across the lifecycle. The following sections describe information modeling techniques and methodologies with which Data Administrators should be familiar.

5.3 Entity-Relationship-Attribute Model

The Entity-Relationship (E-R) approach to information modeling was initiated in the mid-1970s [CHEN76]. In the 1980's, the E-R model was extended for a variety of purposes [TEOR86], and came to be called the Entity-Relationship-Attribute (E-R-A) model.

In this approach, individual items of information are referred to as "entities." Entities are similar to nouns and are used as proper names to refer to particular information items.

Like nouns, entities can represent a wide variety of information items such as data items, functions, and activities.

The associations among entities are referred to as "relationships." Similar to verbs, relationships are used to establish cross-referencing associations among entities.

Descriptive terms are referred to as "attributes." Used similarly to adjectives and adverbs, attributes contribute additional information. Originally used only to describe entities, attributes later came into use as descriptive terms that could be applied either to entities or to relationships.

Entities, relationships, and attributes are the particular values defined for instances of an E-R-A information model. The structure of an E-R-A model is defined by more general classes of information. An "entity-type" is a term representing a general class of entities; a "relationship-type" is a term representing a general class of relationships; an "attribute-type" is a term describing a general class of attributes. Class "types" are referred to below when discussing extensions to the E-R-A model.

5.3.1 Binary Relationship Model

The binary relationship model represents the type of relationships used in the original E-R model. Binary relationships are verbal phrases used to establish an association between two entities. A relationship used in a binary model must always associate exactly two entities. The binary relationship can be defined as either single-directional or bi-directional.

For example, if one entity-type is ELEMENT and another entity-type is PROCESS, they can be associated in the single-directional binary relation ELEMENT-IS-USED-IN-PROCESS. A binary relationship-type can be used to represent both the statement ELEMENT-IS-USED-IN-PROCESS and its reverse, PROCESS-USES-ELEMENT. The combination of these two statements is a bi-directional relationship-type. Binary relationships are defined by both these single-directional and bi-directional relationship-types. To simplify illustrations for the purposes of this guide, only single-directional relationships are shown.

Figure 7 illustrates two simple example binary E-R relationships. This depiction is insufficient if the user is concerned that the same EMPLOYEE-NAME displayed on the screen PERSONNEL-SCREEN is the same EMPLOYEE-NAME used in the process PROJECT-ASSIGNMENT. In this example, the user cannot reconstitute the contextual association linking the three entities EMPLOYEE-NAME, PROJECT-ASSIGNMENT, and PERSONNEL-SCREEN. While all the many entities related to EMPLOYEE-NAME can be recalled, the particular context of this association of these three entities cannot be restored.

Although this simple example may not point up the effect of this limitation of the binary model, the limitation would become obvious in an attempt to model a dataflow diagram, in which a piece of data flows from a process and to a data store. In such a case, the relationship of all three entities must be known to capture the dataflow. This limitation of binary relationships has led users to adopt inventive data modeling devices to capture the context of such a multiple relationship.

An example of such an inventive data modeling device is illustrated in figure 8. The entity-type ELEMENT-IN-CONTEXT permits the definition of a unique entity, EMPLOYEE-NAME-DISPLAYED-FOR-PROJECT-ASSIGNMENT, which is used only to establish this particular association with the other three entities. Because this singular entity is not used in any other context, it creates a unique bond between the entities EMPLOYEE-NAME, PROJECT-ASSIGNMENT, and PERSONNEL-SCREEN.

This method provides users of binary modeling tools with a means to capture more complex relationships among entities. As you can see, however, this method of using a unique entity to define a complex relationship is rather awkward and time-consuming. The n-ary model provides a simpler, more elegant solution.

5.3.2 N-ary Relationship Model

The n-ary model is an extension of the E-R-A model to encompass relationships that join more than two entities. The "n" in the term "n-ary" refers to any number greater than two. When highly complex information models were developed, the binary model for relationships was found to be inefficient. The n-ary model was designed to support the association of multiple entities in a relationship. An example n-ary relationship is shown in figure 9.

For example, if one entity-type is ELEMENT, a second entity-type is PROCESS, a third-entity type is SCREEN, an n-ary relationship-type could be created with the relationship term USE or IS-USED-BY. Using these expressions, an n-ary relationship-type could be defined to represent ELEMENT-IS-USED-BY-PROCESS-AND-SCREEN. This n-ary relationship fully captures the unique association among the entities EMPLOYEE-NAME, PROCESS, and PERSONNEL-SCREEN.

This n-ary relationship-type could also be defined to include the reverse expression PROCESS-AND-SCREEN-USE-ELEMENT, to convey similar information.

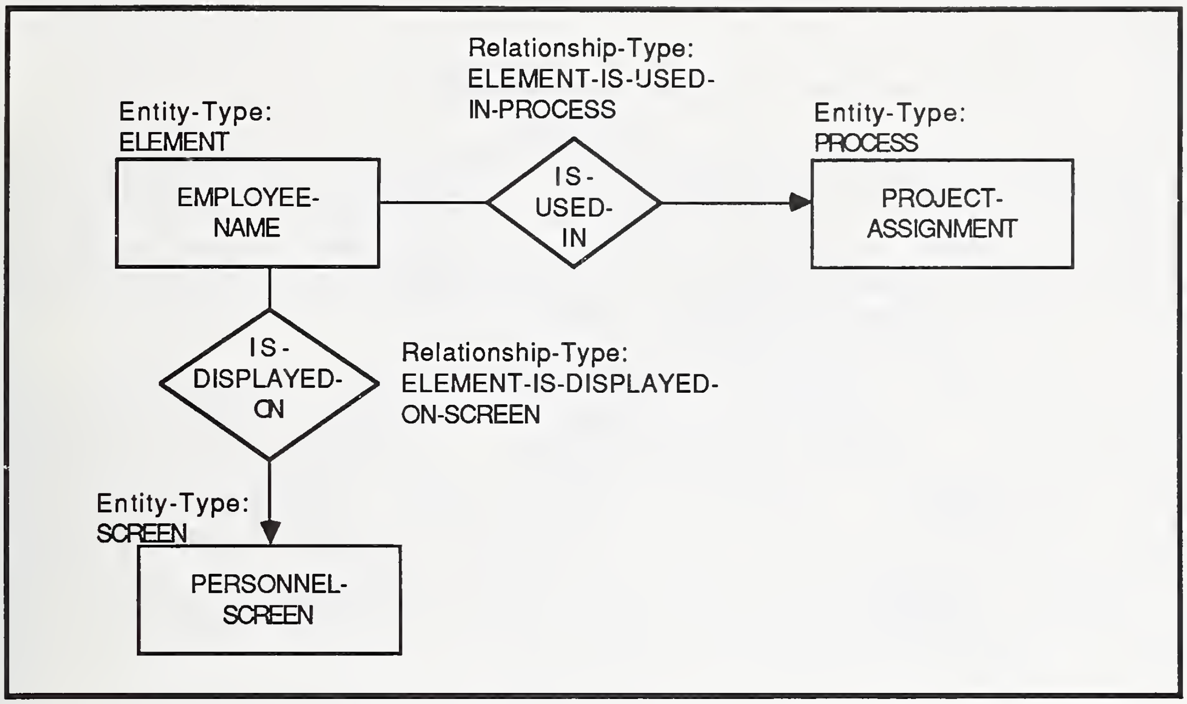


FIGURE 7. EXAMPLE OF INSUFFICIENT BINARY RELATIONSHIPS

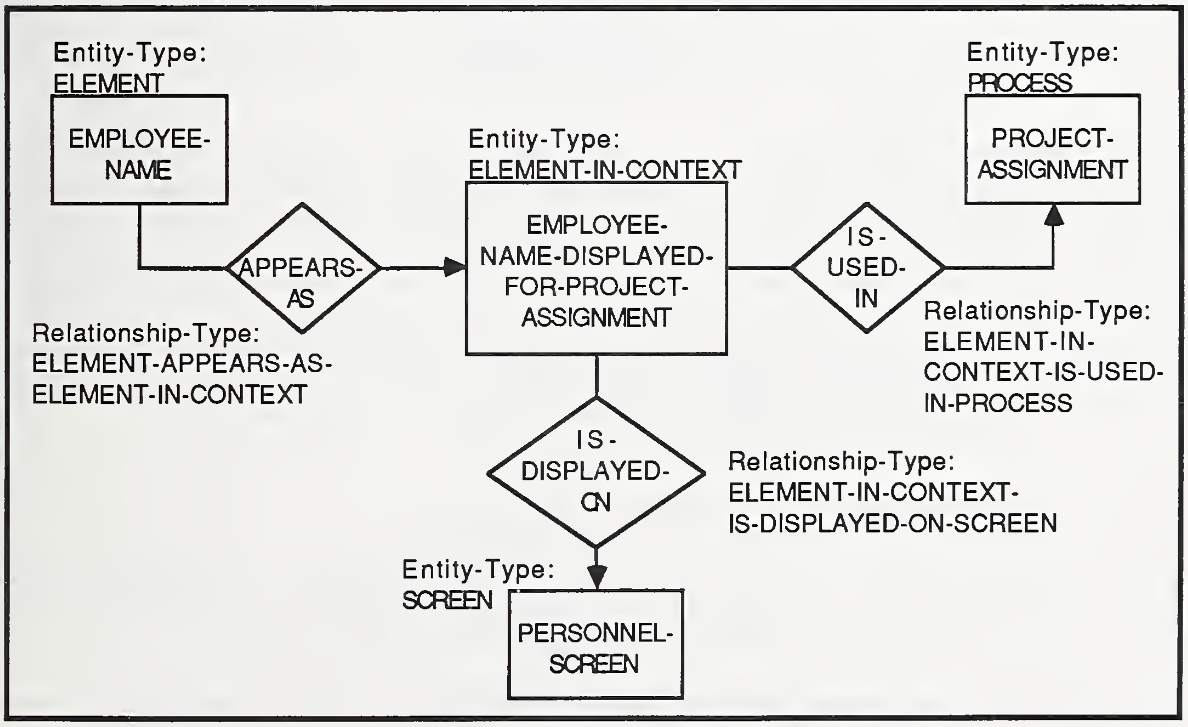

Entity-Type: ELEMENT
EMPLOYEE-NAME
Entity-Type: ELEMENT-IN-CONTEXT
Entity-Type: PROCESS
PROJECT-ASSIGNMENT
APPEARS-AS
EMPLOYEE-NAME-DISPLAYED-FOR-PROJECT-ASSIGNMENT
IS-USED-IN
Relationship-Type: ELEMENT-APPEARS-AS-ELEMENT-IN-CONTEXT
Relationship-Type: ELEMENT-IN-CONTEXT-IS-USED-IN-PROCESS
IS-DISPLAYED-ON
Relationship-Type: ELEMENT-IN-CONTEXT-IS-DISPLAYED-ON-SCREEN
Entity-Type: SCREEN
PERSONNEL-SCREEN

FIGURE 8. EXAMPLE OF IMPROVED BINARY RELATIONSHIPS

5.3.3 Relationships-on-Relationships Model

A recent extension of the E-R-A model permits the definition of relationships-on-relationships. Instead of limiting a relationship to an association of entities, this modeling technique allows a direct association between relationships. Like the n-ary model, the use of relationships-on-relationships supports the representation of complex models.

An example relationship-on-relationship is depicted in figure 10. The example begins by showing a simple binary relationship, ELEMENT-USED-BY-PROCESS, linking two entities EMPLOYEE-NAME and PROJECT-ASSIGNMENT. The example continues, however, by showing another relationship extending from the relation USED-BY to the entity PERSONNEL-SCREEN. The complex relationship illustrated can be described by the relationship-type ELEMENT-USED-BY-PROCESS-IS-DISPLAYED-ON-SCREEN. As you can see, this example of relationships-on-relationships captures much the same information as the n-ary model.

5.3.4 Attributes on Relationships

Attributes were originally used to describe only entities. It was found, however, that modeling was awkward without a further means of describing relationships. To provide this means, attributes were extended to describe relationships as well as entities in the E-R-A model.

An example attribute on a relationship is illustrated in figure 11. In this figure where one entity-type is defined as PROGRAM and another entity type is DATABASE, these entity-types can be joined in the binary relationship-type PROGRAM-ACCESSES-DATABASE. To describe the average number of times the PAYROLL-PROGRAM accesses the EMPLOYEE-INFORMATION database in a month, the attribute-type TIMES-ACCESSED-PER-MONTH is defined for the relationship. The ability to define attributes for relationships provides a more flexible and complete means of expression.

5.4 Structured Analysis and Design Models

First developed in the 1970's, structured analysis and design methodologies continue to be widely used to support information systems and engineering systems development.

The major representation technique of structured analysis methodologies is the use of dataflow diagrams to describe the flow of data through a series of transforming processes. The actions performed in processes "transform" data from one form to

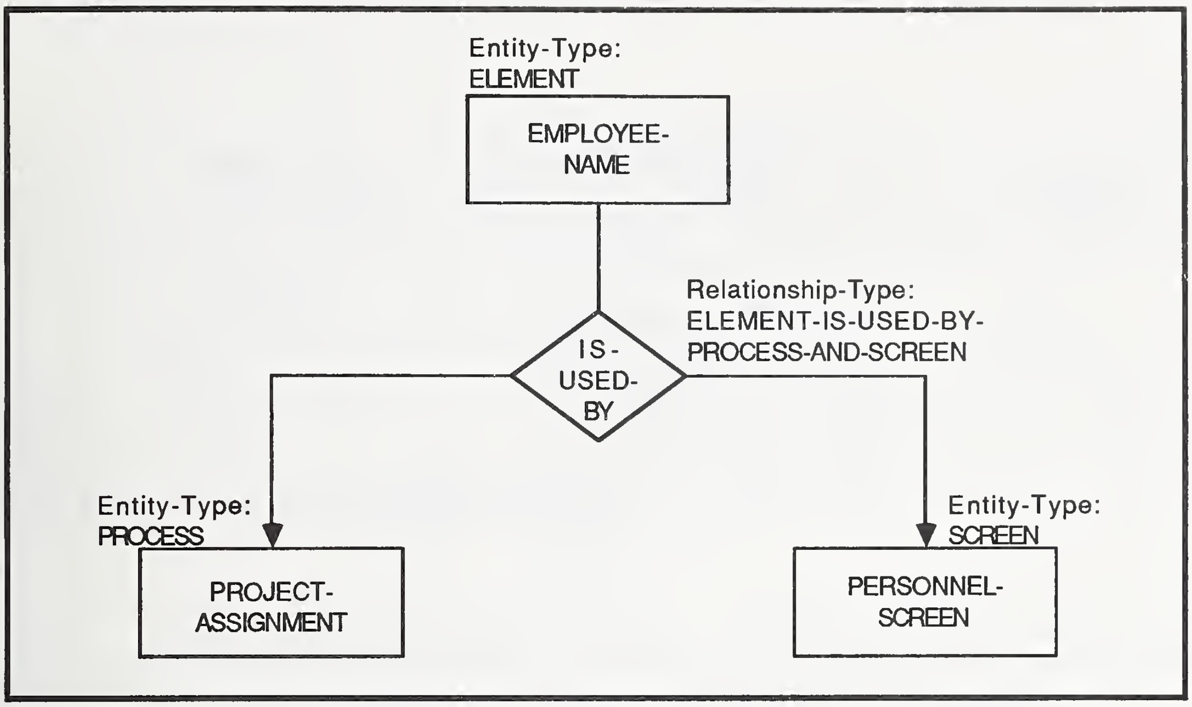


FIGURE 9. EXAMPLE OF AN N-ARY RELATIONSHIP

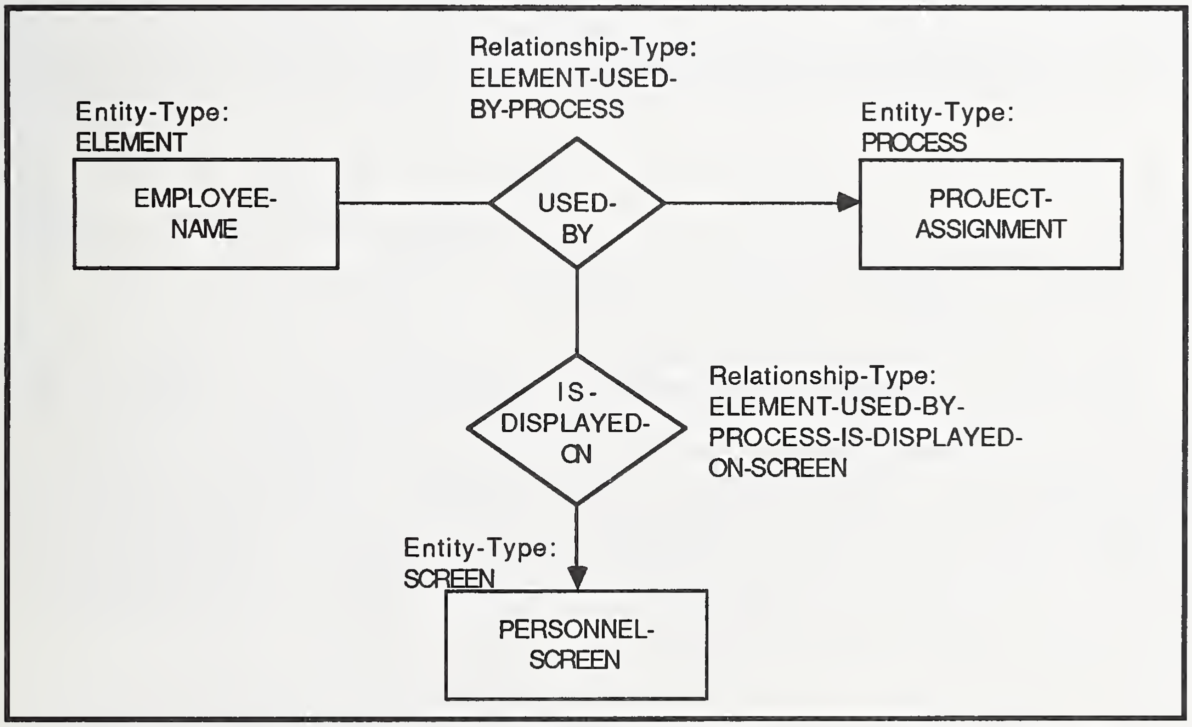


FIGURE 10. EXAMPLE OF RELATIONSHIP-ON-RELATIONSHIP

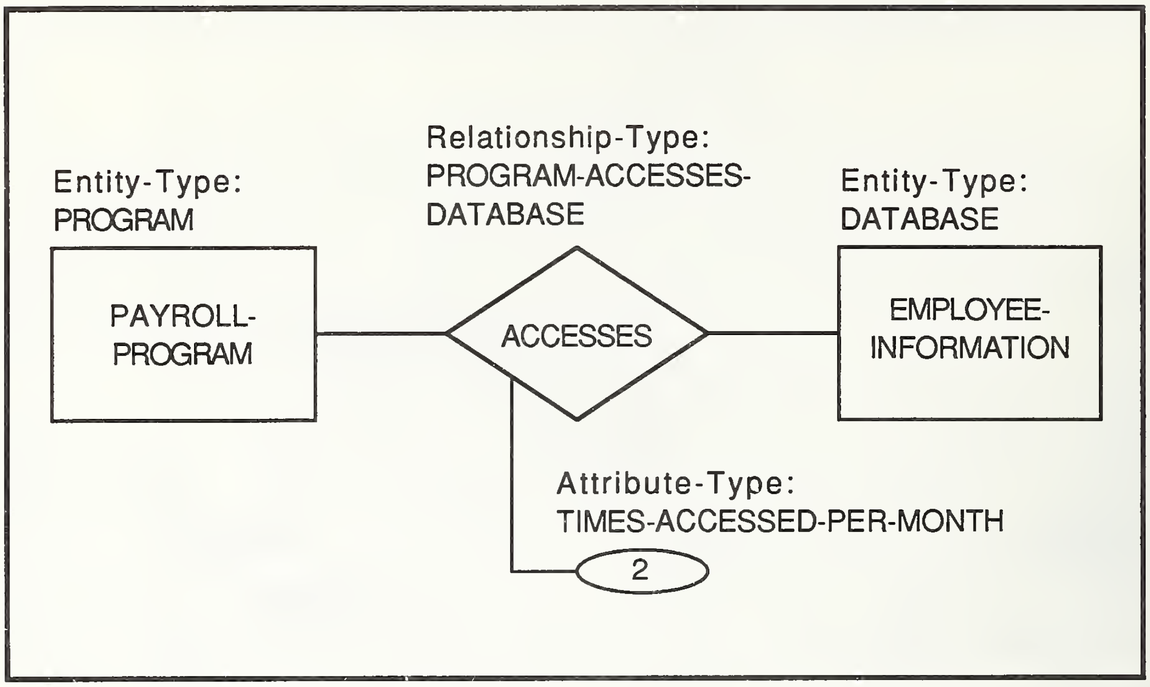


FIGURE 11. EXAMPLE OF AN ATTRIBUTE ON A RELATIONSHIP

another. Dataflow diagrams are also used to describe the access of the transforming processes to shared data stores. Structured methodologies generally use top-down functional decomposition to partition the analysis model [DEMA79]. For further information on structured analysis, see [YOUR89].

An extension of the dataflow diagram technique is used in structured analysis performed for real time systems. In this extension, data transformations are identified by stimulus-response partitioning, that records the events and actions that should trigger particular system responses [WARD89].

Structured design transforms the dataflow diagrams, which were generated during the analysis phase, into a series of structure charts. These structure charts represent the system modules in a hierarchical model that shows parameter passing among a system's subroutines. The series of structure charts establish a hierarchical structure in which the more abstract processing is carried out in the higher level modules, which direct the procedures performed in the lower level modules [PAGE80]. For further information on data analysis and database design, see [PERK84] and [FONG86].

For direction on how to incorporate object-oriented principles with the structured analysis and design methodologies used by many organizations, see [WARD89].

5.5 Object-Oriented Model

First created as a programming discipline, the object-oriented approach is rapidly gaining recognition and popularity. Object-oriented analysis and design techniques have been extrapolated from the object-oriented programming methods, so that the object-oriented approach can be carried out through analysis and design, as well as implementation.

Because the object-oriented approach is relatively new, it is not always as well understood or consistently discussed as other methodologies and modeling techniques. A variety of different terms are used, not always consistently, to explain the object-oriented approach. The following description, therefore, addresses only briefly the most prominent principles of the object-oriented approach.

The object-oriented approach reflects a shift of attention away from program procedure design, and towards the use of data organization to support the definition of program structures and procedures. Object-oriented programming is defined by the ability to distinguish between the general properties of an abstract data type and the properties of a specific data type [STRO88].

For guidance on the principles of object-oriented programming, and on the features required for fully functional object-oriented programming languages, see [STRO88]. For a comparison of three object-oriented requirements analysis (OORA) approaches, see [CHAM89].

5.5.1 Data Abstraction

The object-oriented concept of data abstraction refers to user-defined data typing. By establishing a series of "abstract data types," an analyst or programmer can specify a full set of operations to be performed for each data type.

Data abstraction is derived from the programming practice of strong data typing. In the object-oriented approach, data typing is extended to define the characteristics of user-defined abstract data classes, and to define the inheritance of class characteristics among related data classes.

5.5.2 Data Aggregation

Similarly, data aggregation is derived from the practice of programming in modules. Modular programming leads to the centralization of all data of a certain type under the control of a type-manager module. Data aggregation refers to the practice of grouping related function and data type definitions together in program modules.

The purpose of data aggregation is to logically associate data type definitions with the procedures and functions that use them, and to permit an analyst or programmer to control which data types are accessible by the rest of the program. Also called "data hiding," data aggregation permits data type definitions to be "hidden" in the program modules that use them, so that other program modules cannot inadvertently access those data types. This keeps the program design clean and simplifies software maintenance.

5.5.3 Inheritance

Inheritance is an object-oriented principle for defining data class hierarchies, by which the general properties of a data class are inherited by a subordinate data class. In this approach, inheritance allows the "derived" data class to inherit the properties or attributes of the "base" data class.

This principle of inheritance describes both "simple inheritance," where one data class derives its attributes from one other data class, and "multiple inheritance," where one data class derives its attributes from more than one other data class. In multiple inheritance, a "derived" data class inherits the attributes of multiple "base" data classes.

Use of the inheritance principle in object-oriented analysis and programming permits differentiation of the general properties of an abstract data type from the properties of a specific data type.

6. TOOLS FOR INFORMATION MODELING

Automated support for information modeling is provided by two general classes of tools: (1) data dictionary systems, preferably compliant with the Information Resource Dictionary System (IRDS) Standard that will be discussed in detail in the next chapter; and (2) computer-aided software engineering (CASE) systems. CASE tools, which have gained prominence in the late 1980's, may be based on the same concepts as data dictionary systems. While data management is one of the end results of information modeling, database management systems (DBMS's) are much more appropriate for data management then for information modeling.

6.1 Background of Data Dictionary Systems

Until the early 1970's, all systems development activities were modeled by hand, analyzed through manual comparisons, and recorded on paper. While systems remained small, these manual procedures were cumbersome but adequate. With the advent of large information systems development efforts, however, the tremendous amount of systems development information overwhelmed designers using manual methods. Due to these problems in handling systems development information, automated tools were devised to support the process.

Early systems development activities had been limited to the programming of individual systems, such as the computerization of a company's payroll system. By the late 1960's and early 1970's, data processing had matured sufficiently that larger and more complex systems were computerized. Large system development projects required the management of extensive interrelated system development information, generated for multiple subsystems and multiple lifecycle phases. Systems analysts and designers realized that they needed data management support for the system development process itself.

In the early 1970's, independent data dictionary systems were first designed and implemented. An outgrowth of database technology, independent data dictionary systems were devised to provide an automated means of representing, analyzing, and storing information about the results of the many complex activities performed during the development of large information systems. Many data dictionary systems incorporated prescribed system development methodologies into their built-in procedures and predefined dictionary schemas.

As these data dictionary systems became more widely used for different types of system development projects, their strengths and their weaknesses became apparent. Cross-referencing of information, which was very difficult to achieve in manual systems, was viewed as a strength of data dictionary systems. Users liked the fact that data dictionary systems were able to keep track of the interrelationships among different types of system development data.

The prescribed methodologies and built-in procedures provided by many data dictionary systems were viewed as both a strength and as a weakness. If users were working on a project for which the prescribed methodology was appropriate, the predefined procedures were viewed as a strength. Users tended to follow such prescribed methodologies with enthusiasm bordering on fanaticism. On the other hand, if the same data dictionary system was being used for a different type of purpose, and the methodology was even partially inappropriate, users viewed the predefined procedures as a hindrance.

The schema, or data modeling constructs, provided with the data dictionary system was viewed in a similar light. If the predefined schema was appropriate for the project undertaken, data dictionary system users were enthusiastic. On the other hand, if the predefined schema was insufficient for their needs, data dictionary system users were dissatisfied. In some cases, dissatisfied users would attempt to "kludge" the system by building elaborate data structures to bypass a hindering data dictionary system schema or methodology.

6.2 Features of Successful Data Dictionary Systems

A data dictionary system should be able to support metadata definition, description, and management, including the cross-referencing of information. Schema extensibility, metadata analysis, and easy to use report generation are desirable features for a data dictionary system. The tool should provide guidance to users on how to use the data dictionary system to follow one or more appropriate methodologies. The data dictionary system should, however, be sufficiently flexible to permit users to deviate from the methodology when necessary.

Data dictionary systems should provide support to users for metadata naming analysis and verification. The partitioning of metadata according to lifecycle phases, or other user-specified logical groupings, should be supported by a successful data dictionary system. Data dictionary systems should provide a standard method and provide functionality to support metadata interchange among other repositories.

6.3 Background of CASE Tools

One of the primary weaknesses of data dictionary systems has been the user interface. A data dictionary system has historically been driven by a procedural user interface based on a data dictionary language. The user accesses a data dictionary system through a series of commands issued in the language of that system. Each data dictionary system has its own procedural language, with its individual features and quirks.

Procedural language user interfaces are often difficult to use. A procedural language interface requires the user to be aware of the syntax of the data dictionary system language at the same time that the user is trying to communicate the semantics of a system description, or system model. Also, as each procedural language interface is different, no continuity is provided between different data dictionary tools. Finally, a procedural language interface does not provide the user with an easily understood image of the model the user is trying to produce.

CASE tools were designed to provide a more user-friendly environment for pursuing software engineering. Most CASE tools provide users with a graphics interface, rather than a procedural language interface. A graphics interface is usually designed with a series of surrounding panels that provide the user with functionality similar to that of a procedural language, but with the added capability of being able to view the image of the information being modeled.

While CASE tools often provide a better user interface, some data dictionary system functionality has been lost in some CASE tools. For example, while most CASE tools are able to capture graphical information, not all CASE tools are able to manage the descriptive information that should be captured with graphical objects. The ability to cross-reference different types of information, one of the major breakthroughs of data dictionary systems, is not always a feature of CASE tools. Semantic information analysis capabilities, which were provided with most data dictionary systems, are missing from some CASE tools.

Like earlier data dictionary systems, some CASE tools are still based on a predefined schema structure that users cannot amend. While such predefined schema structures may be helpful to many users, a locked schema limits the uses to which a CASE tool can be applied. Since users' needs for software engineering support are always changing and unique, no fixed schema can ever be sufficient for all software engineering applications. If a CASE tool does not have an extensible schema capability, users of that tool will eventually find it insufficient to their needs.

6.4 Features of Successful CASE Tools

The better CASE tools have adopted the successful features of data dictionary system technology and employ a sophisticated user-friendly graphics interface. The graphics interface of a CASE tool should provide users with the abilities both: (1) to capture semantic information graphically, and (2) to display or reflect the complex interrelationships of semantic information captured in the underlying repository.

Along with a graphics interface, a CASE tool should support the capabilities of semantic information management (including cross-referencing), schema extensibility, and semantic information analysis and reporting. A CASE tool should provide guidance to users on how to use it with one or more appropriate methodologies. CASE support for the methodology, however, should be flexible enough to permit users to deviate from it when necessary.

Since CASE tools are required throughout the system lifecycle, CASE tools should support the partitioning of information according to lifecycle phases, or other user-specified logical groupings. Support for the management of different types of information captured during the lifecycle should be supplied by CASE tools. Like data dictionary systems, a CASE tool should supply a standard method and functionality to support data interchange among other CASE tools and repositories.

While the trend in CASE tool design in the 1980's has been to emphasize the graphics interface features of CASE, this may change in the future. In the 1990's, the trend in CASE tool design may return to greater emphasis on the repository, or data dictionary system, technology. The repository is expected to receive greater attention once again by playing a critical role in information integration and interchange.

In future automated support for software engineering, the repository is expected to be the centerpiece of the software engineering environment, surrounded by a variety of specialized CASE tools that access the repository to manage, integrate, and interchange their data. CASE tools in the 1990's are expected to incorporate additional features, such as programming environments and code generators.

6.5 Tool Support for Data Administration

Data Administration is the management of information describing the functions, operations, and structure of both manual and electronic data processing systems and databases. The Data Administration function is performed throughout the ADP system lifecycle: (1) during the system development phases, (2) during the system operations phase, and (3) during the system maintenance phase.

Since an information system is often comprised of multiple subsystems, which may be undergoing work in different lifecycle phases, the Data Administration of an information system may be involved simultaneously in all phases of the lifecycle.

CASE tools and data dictionary systems provide automated support for Data Administration throughout the system lifecycle. While often associated primarily with system development activities, CASE tools should be able to support the full range of system lifecycle activities.

Instead of providing full lifecycle support, many CASE tools focus on particular lifecycle phases. The majority of CASE tools support the early system development phases, such as Strategic Systems Planning and Requirements Analysis. The mid-level system development phases, such as System Design and

Logical Database Design, are supported by a number of more specialized tools that incorporate specific methodologies supporting these activities. The later system development phases, such as System Implementation and Physical Database Design, however, are supported by only a few, highly specialized tools.

While a number of CASE tools are flexible enough to be used during the entire system development process, relatively few CASE tools presently address the associated areas of system operations and maintenance. Data dictionary systems are often used by Data Administrators to document: (1) the system development process from strategic systems planning through system design; (2) the structure of implemented systems, which may differ somewhat from the system design; (3) the performance of operational systems, such as central processing unit (CPU) workload or communications system throughput; (4) scheduled and unscheduled maintenance procedures; and (5) modifications to operational systems that are planned or have been performed.

7. INFORMATION RESOURCE DICTIONARY SYSTEM (IRDS)

The IRDS Standard is a set of software specifications for a standard data dictionary system. The IRDS Standard establishes the requirements for a software tool that can be used to describe, document, protect, control, and enhance the use of an organization's information resources. Specifically designed to support Data Administration as well as related Software Engineering procedures, the IRDS is the result of government and industry efforts to improve the functionality and utility of data dictionary systems.

The IRDS Standard was approved by the American National Standards Institute (ANSI) in 1988, and is published as ANSI Standard X3.138-1988. Following ANSI approval, the IRDS was adopted as Federal Information Processing Standard Publication (FIPS PUB) 156. The FIPS PUB refers to the more complete IRDS specifications provided in the ANSI publication. For a more detailed account of the IRDS Standard, see [GOLD88]. For an extended discussion of IRDS applications, see [LAW88].

7.1 Background of the IRDS Standard

The National Institute of Standards and Technology (NIST, formerly the National Bureau of Standards) has actively participated as a member of the technical subcommittee for Information Resource Dictionary Systems (X3H4). X3H4 is part of the X3 Accredited Standards Committee for Information Systems which operates under the procedures of ANSI.

In the early 1980's, NIST conducted surveys of Federal government Data Administration and other personnel to record users' views of the limitations of the data dictionary systems then available. NIST also asked these same government workers about the types of functionality they would ideally like to have in a data dictionary system.

With the results of those surveys in hand, NIST personnel devised a list of features that should be included in a data dictionary system to provide full support for Data Administration and related Software Engineering activities throughout the system lifecycle. This list of desired data dictionary system features [KONI81] was the starting point for the IRDS Standard.

7.2 IRDS as a Modeling Tool

The IRDS is designed to support information modeling through the information system lifecycle. Schema extensibility is an important feature of the IRDS that permits the expression of a wide variety of user-defined information models. For a detailed example of schema extensibility, the reader should refer to a discussion of IRDS applications provided in [LAW88]. This guide demonstrates how the IRDS can be used to produce a Strategic Systems Planning model for an organization, and how that model can be used to support the goals of the organization.

The IRDS provides a standard for the data dictionary system (or repository) software underlying most CASE tools. Ongoing IRDS standardization efforts are expected to influence CASE tool development in the future, especially in the areas of the IRDS Services Interface and the Export/Import Facility. The IRDS Services Interface and the Export/Import Facility will provide two methods for the interchange of information among IRDS and CASE tools.

7.3 IRDS Family of Standards

The IRDS is based on the Entity-Relationship-Attribute model, described previously. An entity is any person, place, thing, concept, or event about which data is or can be recorded. A relationship is a directed association between entities. An attribute is a property of either an entity or relationship. IRDS use of the E-R-A model is defined in terms of data "typing." The structure of each IRDS application, known as an Information Resource Dictionary (IRD), is defined in terms of entity-types, relationship-types, attribute-types, and attribute-group-types. The definition of these "types" in an IRD schema provide the structure on which the data contents of the dictionary will be based.

7.3.1 Schema Extensibility

One of the most important aspects of the IRDS Standard is a feature called "schema extensibility." An IRDS must be able to support schema extensibility for each IRD application, so that the IRD schema for each application can accommodate the "types" of entities, relationships, and attributes needed to support the data for that particular application.

The IRDS has been designed to support schema extensibility and thereby provide a neutral framework for the expression of a wide variety of user-defined information models. This means that the Dictionary Administrator of an IRD application (running on an IRDS-compliant system) can define the schema for "types" of information to be stored in an IRD. It is up to the Dictionary Administrator of an IRD, and the IRDS users who will utilize that IRD, to define the appropriate framework, or IRD schema, into which to load the information about their organization and its resources. With this schema extensibility feature, IRDS compliant tools can satisfy the needs of many types of organizations.

7.3.2 Command Language, Panel, and Services Interfaces

The IRDS Standard currently specifies two types of user interfaces. The Command Language Interface permits IRD applications on the IRDS to be accessed through a set of procedural commands. The IRDS Command Language can be used either interactively or in batch mode.

The Panel Interface permits IRD applications on the IRDS to be accessed through a series of "panels" or screens. The Panel Interface is designed to provide the same full functionality as the Command Language. The Panel interface may be implemented through the use of screens, menus, and windowing.

A third type of interface, under development in X3H4, is the Services Interface. The Services Interface defines the needed functionality for communications between the IRDS and other software, such as a DBMS. This interface will provide the support required to use the IRDS in an "active" mode with a DBMS, or other types of Data Administration support tools. In an "active" mode, the IRDS will be able to interact directly with other software.

7.3.3 Export/Import Facility

The IRDS standard currently contains an incomplete IRDS Export/Import facility called the IRD-IRD Interface. The technical subcommittee responsible for the IRDS standard, X3H4, has defined an interchange file format to extend the capabilities of the IRDS Export/Import facility. When approved and implemented, the IRDS Export/Import File Format, specified with the Open Systems Interconnection (OSI) communications protocol Abstract Syntax Notation One (ASN.1), will be able to support information interchange among IRDS tools as well as among CASE tools that can use this interface.

X3H4 is also extending the functionality of the IRDS Export/Import facility. This additional IRDS functionality would extend data interchange capabilities to support schema and subschema interchange, and to support audit trail checking to support data concurrency among multiple IRD applications. The use of the Export/Import File Format for information interchange among CASE tools as well as IRDS tools is also being addressed.

The Export/Import task group of X3H4 has completed the draft specification of the Export/Import File Format, and is in the process of devising a draft of the additional IRDS Export/Import functionality. The IRDS Export/Import File Format is being released first for comment while the task group continues work on the additional IRDS Export/Import functionality.

When completed, the IRDS Export/Import module will specify the functionality for: (1) exporting all or part of an IRD schema and application from a source IRD into an intermediate interchange file, (2) checking a target IRD application for schema compatibility with that intermediate file, and then (3) optionally importing the contents of the intermediate file into the target IRD.

IRDS can be used to trace and interchange information between different models describing an organization's information resources. For example, the IRDS will be able to support information traceability and interchange between the Conceptual Schema model, maintained by Data Administration, and one or more Internal Schema models (physical implementation descriptions), maintained by Database Administration. These models may be stored in one or more IRD applications.

To exchange this information, it may be necessary to move data between separate IRD applications. Information interchange of this type is especially important when an organization is operating in one of a variety of distributed database environments. (The different types of distributed database environments are discussed in [FONG88].)

7.4 IRDS and Three Schema Architecture

To maintain up-to-date, computer-analyzable documentation on the information resources of the organization modeled according to the Three Schema Architecture, Data Administration requires a tool that manages descriptive data about information, called metadata (i.e., data about data). The tool designed specifically to support this information is the IRDS.

7.4.1 Modeling the Three Schema Architecture

The Three Schema Architecture, previously shown in figure 3, is also illustrated as an E-R model in figure 12, Alternate View of Three Schema Architecture. In the figure, each of the boxes for the CONCEPTUAL-SCHEMA, INTERNAL-SCHEMAS, and EXTERNAL-SCHEMAS represent a possibly complex model of multiple subparts. The associations between these schema models are depicted with the relationships IMPLEMENTED-AS and VIEWED-AS. These associations each represent a form of mapping or cross-referencing among the schemas.

Figure 12 provides a model showing that an organization's logical view of all its information resources can be centered

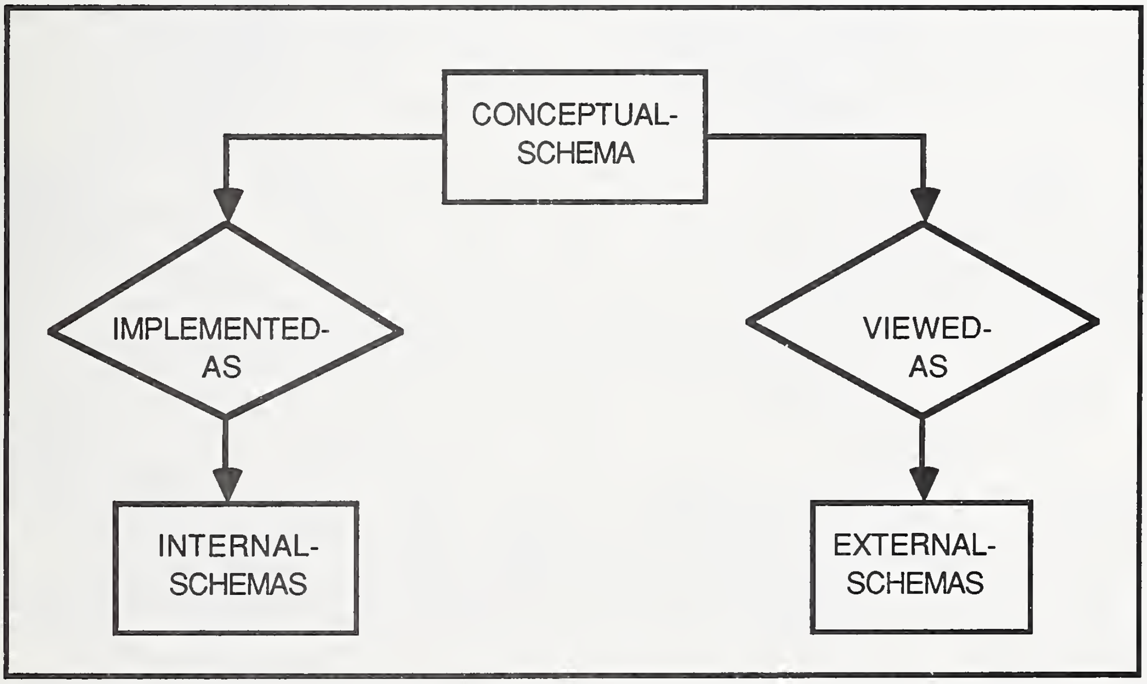


FIGURE 12. ALTERNATE VIEW OF THREE SCHEMA ARCHITECTURE

around its Conceptual Schema. The figure shows that the Conceptual Schema can be implemented as one or more Internal Schemas, each of which provides a physical implementation view of the information resources. The Internal Schemas represent the information stored in computers or files. Similarly, the "big picture" of the Conceptual Schema can be viewed through many External Schemas, each of which provides a limited "user view" of the information resources.

The Conceptual Schema represents a model of the total information resources used in an enterprise. The Internal Schema, or physical implementation model of an information resource, indicates how the information is actually stored in a computer system. The External Schema, or limited user view of the information resources, indicates how an aspect of the information resource is understood by one user group.

7.4.2 IRDS Application of the Three Schema Architecture

The E-R model of the Three Schema Architecture in figure 12 can be extended to be represented in an IRD, or application of the IRDS, through the use of the schema extensibility feature of the IRDS. The model of such an IRD application for the Three Schema Architecture is illustrated in figure 13. For this figure, the terms inside the boxes represent entities, while the terms outside the boxes indicate entity-types that can be defined for an IRD schema. Similarly, the terms inside the diamond shapes represent relationship classes, while the terms outside the diamonds represent relationship-types.

Figure 13 shows how the different types of information resources used by an enterprise can be modeled in terms of the Three Schema Architecture. In this figure, the entity-type CORPORATE-DATA is shown to have one entity instance, NIST-INFORMATION-RESOURCES. This entity-type participates in the relationship-type CORPORATE-DATA-BELONGS-TO-CONCEPTUAL-SCHEMA. The relationship instance shows that NIST-INFORMATION-RESOURCES, an aspect of CORPORATE-DATA, belongs to the CONCEPTUAL-SCHEMA.

The specific entity-types, CONCEPTUAL-SCHEMA, INTERNAL-SCHEMA, and EXTERNAL-SCHEMA, are used rather than the more general entity-type SCHEMA because each schema category will require a different set of attribute-types to describe it. Since an entity-type should be able to share the same set of attribute-types, these schema definitions must each be of a different entity-type.

The entity CONCEPTUAL-SCHEMA is shown to be IMPLEMENTED-AS the PERSONNEL-INTERNAL-SCHEMA, an instance of INTERNAL-SCHEMA.

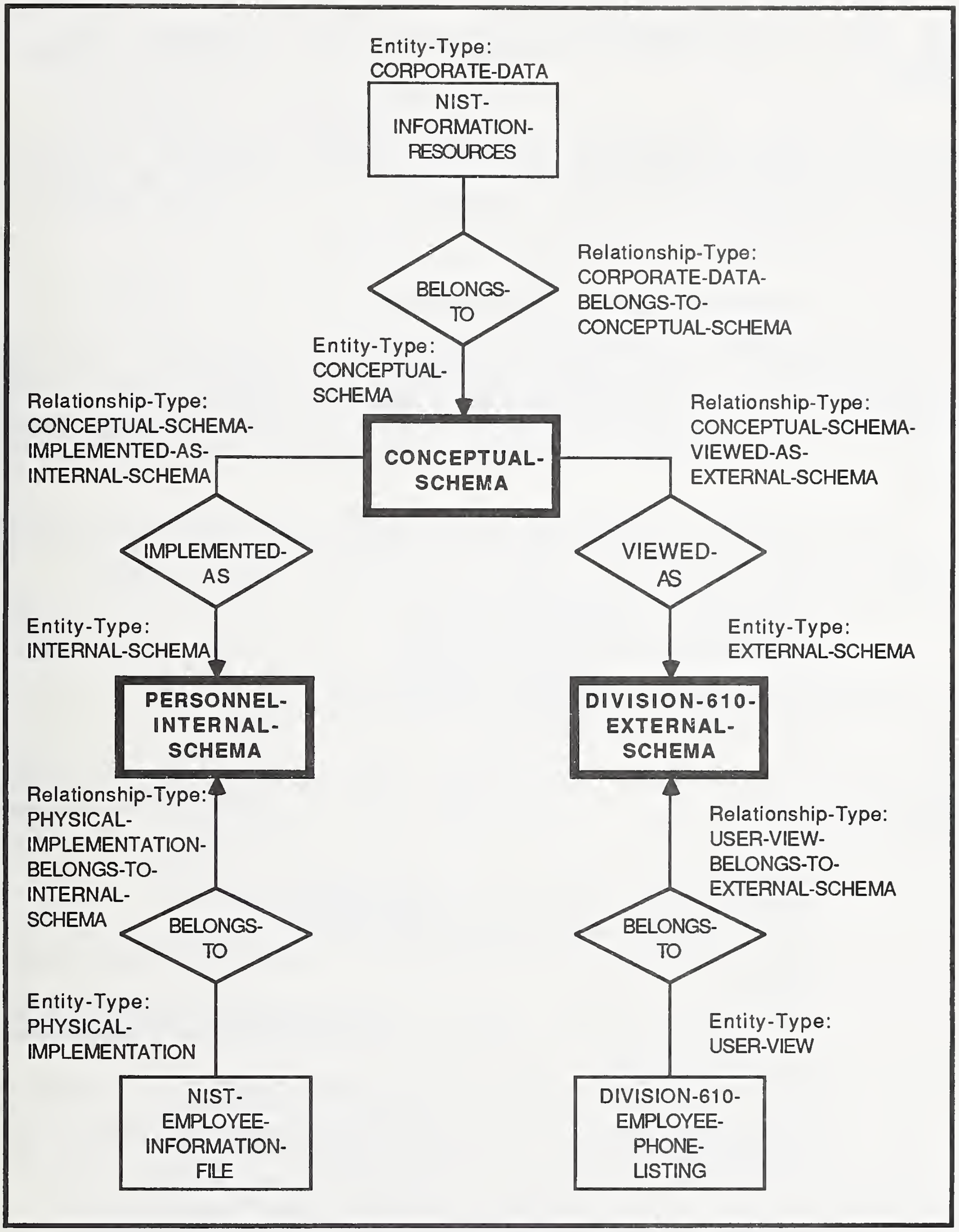

Entity-Type: CORPORATE-DATA
NIST-INFORMATION-RESOURCES
BELONGS-TO
Relationship-Type: CORPORATE-DATA-BELONGS-TO-CONCEPTUAL-SCHEMA
Entity-Type: CONCEPTUAL-SCHEMA
Relationship-Type: CONCEPTUAL-SCHEMA-IMPLEMENTED-AS-INTERNAL-SCHEMA
CONCEPTUAL-SCHEMA
Relationship-Type: CONCEPTUAL-SCHEMA-VIEWED-AS-EXTERNAL-SCHEMA
IMPLEMENTED-AS
VIEWED-AS
Entity-Type: INTERNAL-SCHEMA
Entity-Type: EXTERNAL-SCHEMA
PERSONNEL-INTERNAL-SCHEMA
DIVISION-610-EXTERNAL-SCHEMA
Relationship-Type: PHYSICAL-IMPLEMENTATION-BELONGS-TO-INTERNAL-SCHEMA
Relationship-Type: USER-VIEW-BELONGS-TO-EXTERNAL-SCHEMA
BELONGS-TO
BELONGS-TO
Entity-Type: PHYSICAL-IMPLEMENTATION
Entity-Type: USER-VIEW
NIST-EMPLOYEE-INFORMATION-FILE
DIVISION-610-EMPLOYEE-PHONE-LISTING

FIGURE 13. EXAMPLE IRDS APPLICATION USING THE THREE SCHEMA ARCHITECTURE

And the NIST-EMPLOYEE-INFORMATION-FILE is shown in the relationship class BELONGS-TO with the entity PERSONNEL-INTERNAL-SCHEMA.

Similarly, the entity CONCEPTUAL-SCHEMA is shown to be VIEWED-AS the DIVISION-610-EXTERNAL-SCHEMA. And the DIVISION-610-EMPLOYEE-PHONE-LISTING, an instance of USER-VIEW, is depicted in the relationship class BELONGS-TO with the DIVISION-610-EXTERNAL-SCHEMA.

7.4.2.1 Example IRD Schema Extension

To implement this sample application in the IRDS, first an IRD application is initiated and named. For example, this sample IRD could be called "Three Schema Example." Within this IRD, we must extend its schema in several steps. The first step is to define a number of new meta-entities representing the entity-types.

Following the example shown in figure 13, the entity-types (terms outside the boxes) must be defined as meta-entities:

- entity-type CORPORATE-DATA
- entity-type CONCEPTUAL-SCHEMA
- entity-type INTERNAL-SCHEMA
- entity-type EXTERNAL-SCHEMA
- entity-type PHYSICAL-IMPLEMENTATION
- entity-type USER-VIEW

Next, the relationship-class-types (verbs inside the diamonds) and relationship-types (terms outside the diamonds) are defined as meta-entities:

- relationship-class-type IMPLEMENTED-AS
- relationship-class-type VIEWED-AS
- relationship-class-type BELONGS-TO
- relationship-type CORPORATE-DATA-BELONGS-TO-CONCEPTUAL-SCHEMA
- relationship-type USER-VIEW-BELONGS-TO-EXTERNAL-SCHEMA
- relationship-type PHYSICAL-IMPLEMENTATION-BELONGS-TO-INTERNAL-SCHEMA
- relationship-type CONCEPTUAL-SCHEMA-VIEWED-AS-EXTERNAL-SCHEMA
- relationship-type CONCEPTUAL-SCHEMA-IMPLEMENTED-AS-INTERNAL-SCHEMA

The entity-types in figure 13 are then associated in relationship-types by means of the following meta-entities:

For each entity-type to be associated with another entity-type, the relationship-type is defined with the form:

<entity-type>-BELONGS-TO-CONCEPTUAL-SCHEMA

so that one relationship-type is:

- CORPORATE-DATA-BELONGS-TO-CONCEPTUAL-SCHEMA

Similarly:

<entity-type>-BELONGS-TO-INTERNAL-SCHEMA

so that another relationship-type is:

- PHYSICAL-IMPLEMENTATION-BELONGS-TO-INTERNAL-SCHEMA

A number of relationship-type and other definitions can be made in the IRD schema at this point. IRD schema extension definitions are discussed at length in [LAW88].

7.4.2.2 Example IRD Metadata Definition

When the IRD schema is complete, metadata can be added to populate the IRD application. The entities, shown inside the boxes in figure 13, are added first. This example IRD application is populated with the following entities:

- Conceptual-Schema of entity-type CONCEPTUAL-SCHEMA
- Personnel-Internal-Schema of entity-type INTERNAL-SCHEMA
- Division-610-External-Schema of entity-type EXTERNAL-SCHEMA
- NIST-Information-Resources of entity-type CORPORATE-DATA
- NIST-Employee-Information-File of entity-type PHYSICAL-IMPLEMENTATION
- Division-610-Employee-Phone-Listing of entity-type USER-VIEW

These entities are then related by the appropriate relationships, defined as:

- NIST-Information-Resources CORPORATE-DATA-BELONGS-TO-CONCEPTUAL-SCHEMA Conceptual-Schema
- Conceptual-Schema CONCEPTUAL-SCHEMA-IMPLEMENTED-AS-INTERNAL-SCHEMA Personnel-Internal-Schema
- NIST-Employee-Information-File PHYSICAL-IMPLEMENTATION-BELONGS-TO-INTERNAL-SCHEMA Personnel-Internal-Schema
- Conceptual-Schema CONCEPTUAL-SCHEMA-VIEWED-AS-EXTERNAL-SCHEMA Division-610-External-Schema
- Division-610-Employee-Phone-Listing USER-VIEW-BELONGS-TO-EXTERNAL-SCHEMA Division-610-External-Schema

Each entity added subsequently to the IRD should then be related to the appropriate SCHEMA-LEVEL entity with the corresponding BELONGS-TO relationship.

7.5 IRDS Coordination of Data Administration

As seen in figure 14, an organization's Data Administration policy can be coordinated and implemented through the IRDS. The IRDS can be used as a central repository tool to control a variety of analysis and design tools accessing its centrally stored schemas and metadata. As a central repository, the IRDS should be used to coordinate the development, operation, maintenance, and access control of high-quality data processing systems.

The areas of IRDS coordination support for Data Administration are discussed below. This chapter previously addressed the use of the IRDS to support applications of the Three Schema Architecture.

7.5.1 Name Analysis

When the Naming Convention Verification module is completed, the IRDS will be able to perform name analysis for entities, relationships, and attributes, in order to ensure that all IRD names are consistent with Data Administration directives and dictionary requirements.

7.5.2 Central Repository with Interconnected CASE Tools

When the IRDS Services Interface is complete, and the IRDS can be used in an "active" mode, the IRDS can be used as a central repository in which metadata is stored for use by a variety of tools, such as CASE tools. These CASE tools can support the graphical displays of information, while the IRDS can be used to store and coordinate the descriptive semantic information represented in CASE graphical displays.

The different CASE tools connected to an IRDS central repository should be able to provide a number of specialized services, such as: (1) software module design with support for functional and transform analysis; (2) logical database design with support for reaching the different levels of normal form in relational database design; (3) physical database design with support for implementing databases in a manner most efficient to different types of DBMS products; and (3) application generation

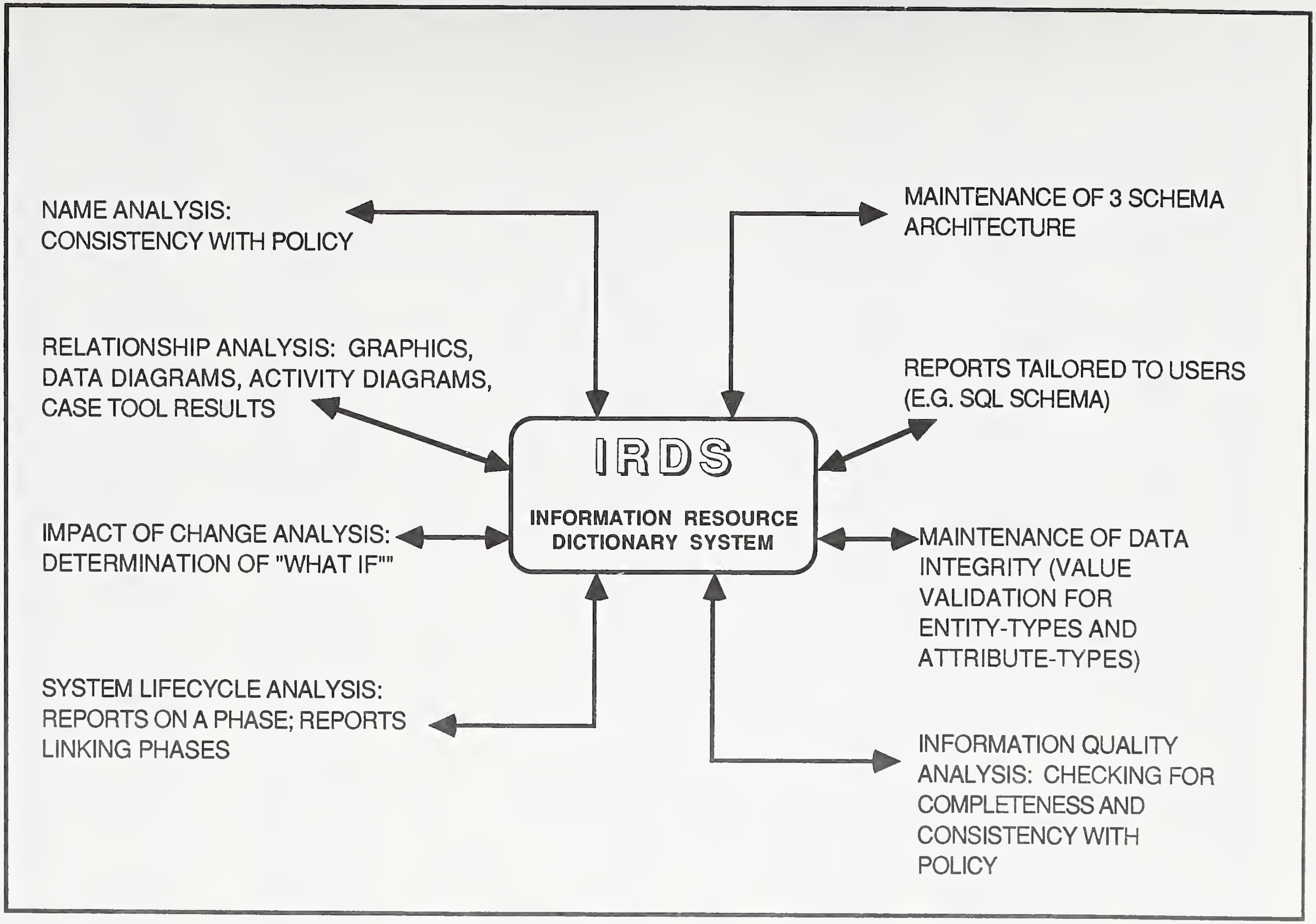

NAME ANALYSIS: CONSISTENCY WITH POLICY
RELATIONSHIP ANALYSIS: GRAPHICS, DATA DIAGRAMS, ACTIVITY DIAGRAMS, CASE TOOL RESULTS
IMPACT OF CHANGE ANALYSIS: DETERMINATION OF "WHAT IF""
SYSTEM LIFECYCLE ANALYSIS: REPORTS ON A PHASE; REPORTS LINKING PHASES
IRDS
INFORMATION RESOURCE DICTIONARY SYSTEM
MAINTENANCE OF 3 SCHEMA ARCHITECTURE
REPORTS TAILORED TO USERS (E.G. SQL SCHEMA)
MAINTENANCE OF DATA INTEGRITY (VALUE VALIDATION FOR ENTITY-TYPES AND ATTRIBUTE-TYPES)
INFORMATION QUALITY ANALYSIS: CHECKING FOR COMPLETENESS AND CONSISTENCY WITH POLICY

FIGURE 14. IRDS COORDINATION AS A CENTRAL REPOSITORY

with support for automatic code generation from system specifications defined using an IRDS.

7.5.3 Information Interchange and Schema Integration

When the extended IRDS Export/Import Facility and the Services Interface are completed, the IRDS will be able to interchange information with a variety of CASE tools. Information interchange, however, is only part of the problem. In order for information interchange to take place, information models of the source and target repositories must be integrated to arrive at compatible schemas.

A number of efforts are now underway in the X3H4 Standards Committee to extend the functionality of the IRDS to enhance its support of Data Administration. Two X3H4 task groups are addressing data interchange topics significant for Data Administration: (1) IRDS Export/Import, to support the data interchange process among IRDS applications and CASE tools; and (2) Schema Integration, to support the integration of differently structured IRDS and CASE applications to be merged through data interchange.

The IRDS Schema Integration task group is considering many scenarios and issues involving the merging of repository applications that have divergent schemas. When the appropriate schema integration techniques are defined and used, the planned IRDS Export/Import Facility and IRDS Services Interface will permit organizations to coordinate the use of a number of analysis and design tools and maintain model integration through the IRDS.

7.5.4 Maintenance of Data Integrity

Two features of the IRDS support data integrity validation within IRD applications. Data integrity at the IRD level also has implications for data integrity in databases. When the IRDS Services Interface is complete, the metadata defined in an IRD application can provide a means for validating the integrity of data in DBMS supported databases.

The Format meta-attribute, at the highest IRDS schema level, allows a dictionary administrator of an IRD to define the Format values permitted for attributes of specific attribute-types. The Format types that can currently be specified in the IRDS are string, text, integer, real, date, and time.

IRDS Attribute-Type-Validation meta-entities, at the highest IRDS schema level, restrict the attribute values that can be defined for particular attribute-types. Validation for "Yes" and

"No" values can be designated through Attribute-Type-Validation-Data. Attribute-Type-Validation-Procedure supports checking for: (1) specific attribute values that can be defined through Value-Validation; and (2) numerical or alphabetical values specified within a range of contiguous values that can be defined through Range-Validation.

7.5.5 Reporting Coordination

The IRDS itself does not, at this time, provide complete analysis and report capabilities to support Data Administration. The IRDS does support Impact of Change reporting that provides information on which entities will be impacted by changes to other entities. Another IRDS reporting function called the Output Syntax, provides IRD application output in the syntax of the IRDS Command Language. The IRDS General Output feature, which permits users to retrieve metadata by selecting different "types" of metadata, provides limited support for user-defined reports.

To provide full support for Data Administration, the reporting facilities of the IRDS should be extended. Other analysis and design tools, with greater reporting capabilities than the IRDS, can be coordinated and used in conjunction with the IRDS, when the IRDS Export/Import Facility or IRDS Services Interface are available.

8. REFERENCE DATA ELEMENT SPECIFICATION

The generic Data Administration function of establishing and maintaining Reference Data Element Specifications is an expansion of the traditional Data Administration function of establishing data element standards.

A reference data element specifies a grouping of informational units. Each reference data element grouping has a unique meaning and has subcategories of distinct units or values. For each unit or value subcategory, the reference data element should also specify the actual values, or types of values, to be used in the organization's databases.

For example, COUNTRIES-OF-THE-WORLD could be a reference data element consisting of various informational units such as access name definition, an alternate name, etc. Also a list of names of countries would be provided as the values (subcategories of distinct units) available under this reference data element (e.g., Afghanistan, Albania, Algeria, Bangladesh, etc.). Finally, a list of the specific representation code assigned to each country would be provided for use in any organizational database (e.g., AF = Afghanistan, AL = Albania, AG = Algeria, BG = Bangladesh, etc.).

A specific reference data element representation can be of any size and can contain the actual attribute values written out, or the reference data element attributes can be represented indirectly through alphanumeric codes. Accordingly, an organization might use a reference data element COUNTRIES-OF-THE-WORLD as a validation check in populating an applications data element known as EMPLOYEE-COUNTRY-OF-BIRTH. The actual values stored in the database could, for example, be either the country names fully spelled out or the two character codes (discussed above) used to represent specific countries, but not both.

By defining and using reference data elements, an organization can develop and refine a vocabulary for use in specifying its information resources.

8.1 Reference Elements and the Three Schema Architecture

When the Three Schema Architecture is used as the basis for tracking an organization's information assets and their use, then it is critical to be able to control the form of data element representations and their mapping to preferred methods of presentation. Without this control, the need for information translators and converters will constantly grow as new external schemas are developed. If users associated with each external schema (user view) then build and maintain their own translation tables, the result is less efficient processing systems.

8.2 Tool Requirements for Data Element Specification

To maximize the benefits of an active reference data element program, Data Administration needs a tool that can satisfy the following requirements:

- Serve as a convenient source of reference data element specifications that can be easily used in developing and implementing system requirements

- Track relationships between reference data elements and actual data elements

- Provide a single point "look-up-table" for translation of physical data element representations to the preferred user representation based upon particular external schemas, or user views.

8.3 IRDS and Reference Data Elements

The IRDS can easily satisfy the requirements for a tool to support a reference data element program. With its extensible schema capability, the IRDS can also provide many additional services useful in defining reference data elements.

For example, as a model is produced for a new organizational process, new reference data elements could be produced that would include multiple relationships. These relationships could be: (1) the relationship between the new reference data elements and the new model; and, (2) the relationships between the new reference data elements and any existing systems that may also employ these new reference data elements. By establishing the relationship between possible new reference data elements and existing systems during the modeling process, it becomes easier for Data Administration to plan an orderly transition from the existing systems to the use of these new data elements.

For another example, Data Administration can use the IRDS extensible schema capability to maintain a record of reference data element conformance. Beginning with the Basic Functional IRD Schema, the predefined ELEMENT entity-type should be used with the addition of a REFERENCE-ELEMENT entity-type. Using these entity-types, a relationship-class-type of CONFORMS-TO should be defined to associate an application data element (ELEMENT) with a reference data element (REFERENCE-ELEMENT). The relationship-type for this example would be ELEMENT-CONFORMS-TO REFERENCE-ELEMENT.

The instances at the IRD application level will define which application data elements (if any) conform to which reference data elements. The definition of attribute-types are useful if the dictionary administrator wants to define in what manner, or to what degree, the application data elements conform to the reference data elements. A data element that conforms to a reference data element should have the same set of attribute-types as the reference data element.

Tracking the use of reference data elements is useful when a system is modified, or when reference data elements are updated. Whenever a system is scheduled to be modified, Data Administration can use the information stored in the IRD to determine which of its application data elements should be changed to conform to the appropriate reference data elements.

9. CONFIGURATION MANAGEMENT

Configuration management (CM) identifies and manages change to the components or items of a functioning unit or system.

Department of Defense (DoD) MIL-STD-490 defines CM as "a discipline applying technical and administrative direction and surveillance to: (a) identify and document the functional and physical characteristics of a configuration item; (b) control changes to those characteristics; and (c) record and report change processing and implementation status" [MILI70].

CM is necessary to ensure change control throughout the system lifecycle. It is needed to manage the continually changing aspect of information system requirements, specifications, and mode of implementation. CM of information throughout system development, during system operations and maintenance, and during system redesign is often the responsibility of Data Administration, working closely with Software Engineering.

Since a completed information system is made up of computer hardware and software, the process of managing change information about computer systems is often referred to as hardware and software configuration management. (For an overview of software configuration management, see [OSBO89]). CM of the products resulting from the system development process is just as complex and important. The IRDS can support CM in all these forms.

9.1 Role of Data Administration in Configuration Management

The generic function of CM applies to many of the activities of Data Administration. In fact, once almost any decision has been made in Data Administration that involves creation or modification of some information asset, the implementation of that decision is in many ways an exercise in CM. The actual establishment by Data Administration of a CM policy is not something that requires any tool support. It is the ability to track activities that are CM related which requires tool support.

The primary requirement of any Data Administration tool used in the CM function is the ability to track any named entity through the various stages of its lifecycle. A further requirement is the ability to maintain the various relationships of entities across different stages in each individual entity's lifecycle. Finally, once a system is in place that can track and retain the needed relationship information, that information could be used to actually implement automated CM controls on the organization's information assets.

Many of the lessons learned in the area of software maintenance control are directly applicable to the CM of information resources. An overview of software CM is provided in [OSBO89] with emphasis on the fact that CM provides a discipline for planning and implementing change control throughout the software lifecycle. [MART83] emphasizes that software maintenance should be performed in a structured, controlled way over the entire lifecycle of an application system and that managers should institute controls over the software process and any changes that are made. The CM lessons discussed in these two documents are also true for information resources. It is critical that the lessons learned in software system maintenance not be forgotten when establishing CM controls for information assets.

9.2 Configuration Management and the Three Schema Architecture

The ability to perform CM is critical when viewed in the context of the Three Schema Architecture. Looking back at figure 4, Different Views of Information Resources, it can be seen that the requirement to change an organization's information assets could come from multiple locations. A user might have a new performance requirement, or Database Administration may request permission to alter physical data storage structures. These changes must be accommodated in the Three Schema Architecture model of the organization's information assets in a manner that does not interfere with the use of actual information assets.

At times, Data Administration will have to keep track of assets at all three levels of the architecture, as they currently exist, as they are proposed to change, and perhaps how these same assets will exist as they move from their current status to the proposed status. Further, Data Administration must maintain all relationships that currently exist, and that will exist as these assets move through their lifecycle. Clearly this is a complex task requiring a tool that can accommodate multiple lifecycle levels and the ability to track relationships across lifecycle levels.

9.3 Configuration Management and the IRDS

The IRDS supports the CM task by: (1) providing extensible schema capabilities so that the dictionary can be tailored to fit each application, (2) capturing descriptive information about configuration items, and (3) supplying user-defined lifecycle phases to partition the information within an application.

Every IRD application of the IRDS must have a structure to capture the types of information with which that dictionary application is to be populated. The underlying structure of an IRD application is its schema. The IRDS provides users with only a limited example schema, the Basic Functional Schema, which is not sufficient to support all the schema structures needed for CM. The extensible schema capability of the IRDS, however, provides users (or more properly, Dictionary Administrators) with the means to design their own schemas, or to supplement the Basic Functional Schema.

To define the schema structures necessary to support a CM application, users should first develop a data model appropriate for the application. A detailed description of the process of developing data models for IRD schemas is provided in [LAW88].

The current version of the IRDS supports three classes of lifecycle phases useful to CM in tracking changes in configuration items as systems evolve.

These three IRDS lifecycle phase classes are Uncontrolled, Controlled and Archived. The Uncontrolled class corresponds to all the system development phases, and any number of user-defined lifecycle phase partitions can be established within it. The Controlled class is a single phase class and it corresponds to the system operation and maintenance phase. The Archived class is also a single phase class, and is used to support the archiving of existing systems as they are replaced by new systems.

The IRDS standard does not require any specific organizational application of these lifecycle phases. They are provided for use only if they are appropriate to the procedures of the organization. The following is an example of Data Administration's use of these three lifecycle phase classes to manage data about a system under development.

Assume a new system is being developed that requires some new information resources, and will also use some previously existing information resources. The new system could be modeled under the IRDS, using the appropriate entities and relationships within the uncontrolled lifecycle phase.

This new system model would have relationships between entities in the uncontrolled lifecycle phase, and, since it would be using some existing information resources, it would also have relationships between entities in the uncontrolled lifecycle phase and the controlled lifecycle phase. The model of the new components of the system would be maintained under the uncontrolled phase since they would likely be subject to modification by multiple sources.

The entities describing already existing information resources to be utilized by the new system would be protected from change since they would be in the controlled lifecycle phase. This ability to maintain relationships between entities in different lifecycle phases encourages the use of already existing information resources.

As the various segments of the new system become fixed, the appropriate entities in the uncontrolled lifecycle phase could be moved to the controlled lifecycle phase. Even after being moved to the controlled lifecycle phase, these entities would continue to have appropriate relationships to other entities that are in the controlled and the uncontrolled lifecycle phases.

Finally, once the entire new system is fully operational, all entities and relationships that make up the system model would be moved to the controlled lifecycle phase.

Optional Module 4 of the IRDS supports integrity rules concerning the transfer of information between lifecycle classes. These integrity rules control metadata transfer: (1) from the Uncontrolled class (containing system development lifecycle phase information) to the Controlled class partition (containing system operations information); and (2) from the Controlled class partition to the Archived class partition (containing archived information describing previous operations).

10. DATA PROTECTION POLICY

While Data Administration has often been concerned with ensuring the confidentiality, reliability, and integrity of information, it was not usually directly involved in system security. The protection of the computer system was considered to be the job of the "security group" of the organization. The security problem was usually handled through the implementation of sign-on control, such as password or more elaborate lock-and-key protection systems, to restrict system access only to designated individuals.

In today's environment, Data Administration must take a more active role in supporting system security. Data Administration should be directly involved in the planning and implementation of security controls. The same procedures and tools that Data Administration should use to ensure the confidentiality, reliability, and integrity of information resources can also be used to enhance system security.

It is the responsibility of Data Administration to ensure that system security controls operate in conjunction with the controls implemented to ensure the confidentiality, integrity and reliability of the information resources of the organization. Further, these system security controls should not be stand-alone modules, but should be integral parts of the information architecture.

The archiving of database and dictionary information contributes to protecting and maintaining historical records of data and systems evolution. Data Administration should play a key role in archiving databases and the dictionary applications that describe the data and systems of an organization's information resources. Archiving both the schema (i.e., structure) and the data content of a database has been a problem. A method for archiving databases, including the database schema and its data content, is described in this chapter.

10.1 Confidentiality, Integrity, and Reliability Policy

Data Administration should play a key role in an organization's system security by contributing procedures and tools to ensure the confidentiality, reliability, and integrity of an organization's information resources.

Data Administration's concerns for data **integrity** require that data instances can be verified to conform to the specific values, or set of permitted values, designated for their data types. These cross-referencing checks to verify correct or permitted data values, and to identify those data values that do not conform, must be implemented with automated tools, such as the IRDS. The IRDS can be used to enforce metadata verification within its dictionary applications, and, with the Services Interface, it can be used to support data verification for other information resources.

Even if the data in a system is verified to be correct when the data enters the system, it still must be protected from unauthorized access and alterations while in the system. Similarly, since systems can be used to access information, systems must also be protected from unauthorized access and alterations. Providing adequate access control to ensure information **confidentiality** is one of the major responsibilities for Data Administration.

Data Administration's concerns for confidentiality are complicated by the requirement to allow **multilevel access** to information resources. Multilevel access is when some, but not all, of the data within a system is to be accessible by a user. Multilevel access can be restricted at many different levels. The multilevel access problem can be complicated further by the requirement for **multiple access types**, such as "read," "write," "execute," "append," "modify," "delete," and "create." When multiple access types are required within a multilevel access system, a user can be assigned different access types for different types of data within one system.

There are many possible scenarios describing the need for multilevel access. In one case, a user might be allowed access to one or more of an organization's databases on one computer, but not to all of the databases on that computer. In second case, a user might have access to some of the information in a database, but not to all of the information in that database. In a third case, a user might have "read and write" access to certain portions of a database, have only "read" access to other portions, and have no access to yet other portions.

Not only must information be protected from intruders, information must also be protected from "valid system users" who are allowed access to some but not all information provided by a system. The same principles of access control applied to a database or data dictionary system can also be applied to a system. Tools and policies to support Data Administration can keep track of information access permissions all the way down to the individual data element level. These same tools and policies can be applied to system security to enforce multilevel system security access.

To perform its confidentiality, integrity, and reliability functions, Data Administration will have to establish certain controls. These controls will be expressed by a set of explicit rules which cover in detail such areas as:

- **Access control** - the process of limiting access to the resources of an ADP system only to authorized users, programs, processes, or other networked ADP systems.

- **Backup and recovery** - a computing facility intended to provide a service in the event of loss of that service from another computing facility; user or operating system actions taken to restore a computer system to working order after a failure.

- **Data Concurrency** - a condition in which the currency and consistency of data content is simultaneously or continually maintained among two or more collections of data sharing a set or subset of data items.

- **Inter- and intra-record constraints** - levels of access control maintained to restrict information access to certain designated records (at the record level), and to restrict information access to certain designated data items within a record (at the data element level).

Since a major purpose of security rules is to control access to the organization's information resources, it is imperative that a method is specified to maintain and enforce the rules. This method must also allow the retention of the relationships between the rules and the information resources they protect.

For example, assume there is a security rule stating that a particular set of data elements can only be accessed by a single system. If that system is removed due to obsolescence, then the subject rule should also be eliminated. Without a tool to maintain the cross-referencing relationships among the security rules, the data elements, and the information resources, these untraceable security rules would become unenforceable and obsolete. Further, as the collection of these obsolete security rules continues to grow without automated support, they could introduce conflicts within the set of all security rules along with causing the entire set to become overloaded. Automated support is required to maintain and enforce security rules.

Chapter 10 briefly addresses only some of the security aspects that must now be the concern of Data Administration. Many of the concepts and methodologies that can be used to help provide adequate confidentiality, integrity, and reliability for information resources are not new to Data Administrators. However, many of the concepts and methodologies used in providing security protection may be new to Data Administrators, although they are not new to security specialists. Accordingly, it is imperative that Data Administration personnel become familiar with many areas of data processing security in order to protect the information assets under their control. The following is an initial list of publications discussing security: [FIPS76], [FIPS77], [FIPS79], [FIPS80], [FIPS81], [RUTH88], and [NEUG85].

10.1.1 Risk Management

Data Administration's efforts to implement security controls to ensure information confidentiality, integrity, reliability should take into account the concept of **risk management**. Risk analysis evaluates the entire spectrum of a system's assets and vulnerabilities to define the degree of risk associated with possible events that may threaten the system. With this risk analysis in hand, risk management defines the physical, technical, and administrative controls and procedures required to develop **cost-effective solutions** for security. Like any project, the development and implementation of security controls will be constrained by the amount of resources available.

To enhance security, Data Administration may wish to participate in establishing a certification function, which involves independent verification of a system's risk management. Certification is part of the accreditation process performed when a requirement exists to ensure that a particular computer or network system meets a prespecified set of security requirements. Accreditation is the authorization and approval granted to an ADP system or network to process sensitive data in an operational environment [FIPS76]. For more information on computer security certification and accreditation refer to [FIPS102].

10.1.2 Maintaining Security Access Rules

Following IRM policy security policy, Data Administration should ensure adequate confidentiality, integrity, and reliability protection for the information maintained in the information resources of an organization. Accordingly, Data Administration should establish a set of **rules and procedures** to provide the needed levels of protection. While some portion of these rules might apply to the initial act of identifying valid users of the system, the requirements for these rules will extend well beyond this initial identification operation.

Once an individual is granted initial access to the organization's information processing facilities, that person's information access must still be under some form of control. Access must only be allowed to that information for which the individual has previously been identified as having a legitimate access requirement.

A set of rules and procedures should be maintained to control access to specific information resources. Often, an integral part of these rules and procedures will be in the form of a **permission list**. Organized according to individual, project group, or some other category, this permission list should specify what information resources each individual is to be allowed to access, and the level of access permission to be allowed within each information resource. In this role of supporting information confidentiality, integrity, and reliability, Data Administration must perform maintenance and protection of these access rules and procedures, along with their associated permission lists.

10.1.3 Storing and Managing Access Rules

The IRDS is one possible mechanism for storing and managing the access rules, permission lists, lists of data elements, lists of information resources, and their cross-referencing relationships. By using the IRDS for this purpose, Data Administration can retain the rules and lists along with their associations with the various entities and relationships that make up the External, Conceptual and Internal schemas of the Three Schema Architecture. This concept does, however, place a security requirement on the IRDS. If used to maintain the security and integrity of these rules and lists, the IRDS must have some form of built-in protection that prevents unauthorized access to its contents. This security requirement is not unique to the IRDS.

During the development of the IRDS standard, X3H4 recognized that the IRDS could provide an extremely detailed, descriptive "picture" of the information assets of an organization. Since it would be desirable for an organization to have the capability to restrict access to portions of this "picture," it was decided that a security requirement for the IRDS did exist. In order to address this requirement, Optional Module 3 of the IRDS, known as the Security Module, was developed.

The **Security Module of the IRDS** was designed to restrict access to specific information depending on the access type specified for each user. Access restriction can be performed on both the dictionary schema and the dictionary content; it can be enforced for specific items of information at the entity level, or across an entire dictionary application at the global level.

With the Security Module, the IRDS can be utilized as an **access control mechanism** to detect and prevent unauthorized access, and to permit authorized access to an ADP system. The Security Module can be used to protect the IRDS itself, as well as an access control mechanism to provide multilevel restricted access to other ADP systems among the information resources. A further discussion of the IRDS Security Module is included in [GOLD88].

By utilizing the IRDS Security Module, Data Administration can protect the confidentiality of its information, and maintain the "picture" of the organization's information resources.

10.1.4 Protection Against Unauthorized Access

The first item of consideration in the security areas of confidentiality, integrity, and reliability is the need to prevent access to the information resources by those who have no legitimate requirement to access those resources.

One of the most recent items that has become available for use by Data Administration in limiting access to information is an actual physical tool. This physical tool is called the Smart Card. A **Smart Card**, when used in conjunction with appropriate software, can be used to help verify the identity of an individual attempting to access the organization's data processing system. A Smart Card is:

> ". . . a credit-card-sized device containing one or more integrated circuit chips, which perform the functions of microprocessor, memory, and an input/output interface" [HAYK88].

As a simple example use of a Smart Card, assume an individual's card contains unique identification information about its authorized holder (e.g., mother's maiden name, name of high school, etc.). Individuals using the Smart Card for identification can be restricted from access to the information it contains. Each of the unique pieces of identification information on the Smart Card can be used to verify access to one or more specific portions of the organization's information resources. This access relationship could be maintained in the organization's IRDS.

As an example of **authorization verification** with a Smart Card, the following scenario could occur. First an individual uses a Smart Card when attempting to sign on a system protected from unauthorized access. The system asks the individual to verify one or more pieces of the identification information from his or her Smart Card. If the individual responds correctly, he or she is allowed initial access to the system. The individual then asks for a retrieval from one of the databases on the system. The system checks the access control rules defined in the IRDS to verify whether the individual is authorized to access that database and that type of information.

At this point, the access control rules in the IRDS could prompt the system to ask the individual to verify some additional Smart Card identification information. Only when the individual has properly verified the requested identification information would he or she be granted access to the requested database and to the requested type of information within the database.

If the requested identification information did not exist on the Smart Card, or if the individual did not verify it properly, then access would be denied. Thus if a user attempted to view a database or type of data for which he or she is not authorized access, the system would not allow that retrieval. Yet, if that user is an authorized user of other databases on the system, that individual could still have access to those other databases for which he or she has been authorized to access. Additionally, if someone else attempted to use a designated individual's Smart Card, the system would have several levels of protection to prevent unauthorized access.

The above example is one possible elementary scenario for the use of Smart Cards to protect information resources. A more complex system might have codes on the Smart Card that would not only aid in identifying the individual, but would also limit the individual to only accessing particular systems from certain physical devices or locations.

By directing the integration of Smart Card use into the design of new or revised databases, Data Administration is presented with a tool that could help greatly in the protection of the data assets under its control.

10.1.5 Problem of Multilevel Access

The problem of controlling information access becomes more complex based on the level of access control required. The possibility exists that some organizations will require multilevel control of their information resources below the global, or database level. Some organizations will actually have to control access down to the specific data element level.

The reason for providing control down to the data element level is to allow multiple users, having different access level permissions, to each have restricted access to particular types of data from the same database. When granted access to a particular type of data, permissions established for each user's access type will determine the nature of the access, such as the ability to "read," "write," "modify," and/or "delete." These same users, however, must be prevented from accessing other information to which they have not been granted access permission, but which is physically resident within the same database.

If an organization requires this level of data access control, and methods for providing this level of control within a single database are not available, then the only alternative is to have multiple, physically separated databases holding similar information but with different degrees of security classification. When the same data is maintained in separate databases, however, data consistency and concurrency becomes a problem. To avoid these problems, the IRDS can be used as an access control mechanism along with other tools, such as the Smart Card, to implement multilevel data access security.

10.1.6 Audit Trail

Another function that must be considered when establishing security rules is the set of those rules that cover audit trails. While many DBMS products maintain some limited form of audit trails, these are usually for the purpose of backup, recovery, and limited DBMS use analysis. For security, however, these limited audit trails may not be sufficient for use in combating such problems as the mosaic effect.

A **mosaic effect** results when an individual can combine multiple separate pieces of information, which he or she may be authorized to access, to produce a "larger picture" of other information which he or she may not be authorized to know. To protect highly sensitive information in situations of top security, organizations restrict individuals to the least amount of information they "need to know" to accomplish their work.

The "larger picture" that an individual could potentially piece together from information he or she is authorized to access could be information which this individual is restricted from accessing. Automated protection against the mosaic effect would require a tool to track an individual's data access requests, to analyze the information retrieved by an individual for a possible mosaic effect, and to limit access permissions before a mosaic effect begins to result.

While an IRDS application schema can be defined to track users' access requests, and while that same application schema can be defined to associate user access requests with a certain pattern of specified requests that would yield a "larger picture," the IRDS does not have the analysis capabilities needed to fulfill this requirement. At this time, no tools are readily available in this area. It could be possible, however, to utilize one of the Artificial Intelligence (AI) programming languages to devise a knowledge based system application for this purpose. By using AI to analyze an extensive audit trail, maintained in conjunction with the IRDS, it might be possible to produce a limited system of mosaic effect protection for one or more of an organization's most sensitive information databases.

10.1.7 Physical Loss of Information Assets

Protection against the physical loss of information assets involves ensuring the reliability of both currently used information assets and historical information assets. The responsibility of Data Administration is to follow IRM policy in this area and to work with Database Administration to develop and review the procedures necessary to insure that adequate **backup and recovery protection** is provided for all information resources. Database Administration is responsible for implementing and administering the backup and recovery measures that protect information assets from physical loss.

Except in a limited number of specialized situations, it should not be necessary for an organization to have a separate backup and recovery system. Instead, the backup and recovery function should be implemented as an integral part of each DBMS used by the organization to hold its information resources. Accordingly, the type of backup and recovery methods selected for an organization should be part of the evaluation criteria used to select any DBMS used within that organization.

10.2 Archiving Policy

Data Administration should establish and apply a policy to control the archiving of the information assets of an organization. The decision as to what information resources of an organization are to be archived could require a rather extensive set of criteria or rules.

The decision to archive information is usually not based on just a specific time frame, but may also involve a derived set of rules based on the historical value of the information to the organization. When the rules governing archiving are complex, it may be necessary to automate the management of these rules in a tool such as the IRDS. With the IRDS, the archiving rules can be defined and cross-referenced, and traceability can be maintained between the rules, the archived records, and the information resources they are intended to protect.

Archiving is further complicated by the rapid changes in storage hardware and software. As hardware and software change, it becomes more and more difficult to access archived information. It is quite common for the original database software, hardware and operating system to be unavailable for use in accessing that information. Thus, even if some needed archived information is physically present on a storage device, the information will be unusable without proper indexing, schemas, and descriptive information. Without sufficient indexing, schemas, and description, the effect is the same as if the information were never saved. Automated tools are necessary to retain the associated indexing, schemas, and description of archived databases.

10.2.1 Archive Tool Requirements

In order to accomplish the generic function of archiving information resources, Data Administration must insure that future use of archived information is possible. This must be done regardless of technology changes. Thus among the requirements for an archiving tool would be:

- Hardware and operating system independence

- Automatic retention of all relationships among data

- Ability to export from, and import to, any other data management system.

While current DBMS systems may have limited archiving functionality, use of the archived data requires use of the same DBMS system. Once an organization moves on to some new DBMS system, it is entirely possible that easy access to all archived data could be lost. Further, as time passes, changes in an organization might also cause changes to the interpretation of the actual meaning of the data in any given database. To retain the meaning of the archived raw data, it is also necessary to retain sufficient "metadata" to provide its full description, or semantic value.

10.2.2 Current Technical Solution

No tool existing today can independently satisfy all of the archive requirements stated above. Combining the functionality available in many DBMS systems, however, with the functionality provided by the IRDS standard, can produce an archiving capability to satisfy the above requirements.

An example of this type of combined functionality is illustrated in figure 15, Database Archiving Model. The three dimensional disk shapes represent stored information, while the lozenge shapes represent activities to be performed. The solid arrows in figure 15 represent the transfer of actual physical data. The "shadow" arrows represent either the act of storing new physical data or the act of referencing existing physical data from the indicated source in order to accomplish the indicated action.

The IRDS Export/Import File Format is referred to in the following description of preparing a database archive package. The Export/Import Facility, which is being defined for the IRDS, will provide a standard means for the interchange of schemas and metadata among IRDSs and IRD applications.

To prepare a **database archive package**, the following actions should be taken:

(1) The data contents of the database (separate from its structure) is recorded in records in a flat file, as shown by the lozenge shape on the left. This flat file representation of the data is then included in the database archive package, illustrated by the large disk shape at the bottom.

(2) The record structure of the data flat file is documented in an Information Resource Dictionary (IRD) application of the IRDS standard, shown by the center lozenge shape. Only indicating how to read an archived data record, this description of the flat file structure should be very simple. An export version of the IRD describing the flat file structure should be generated using the IRDS Export/Import File Format. This IRD describing the flat file record structure is then included in the database archive package.

(3) The database schema structure and other descriptive information about the database is documented in a separate IRD. The use of the database, its reports,

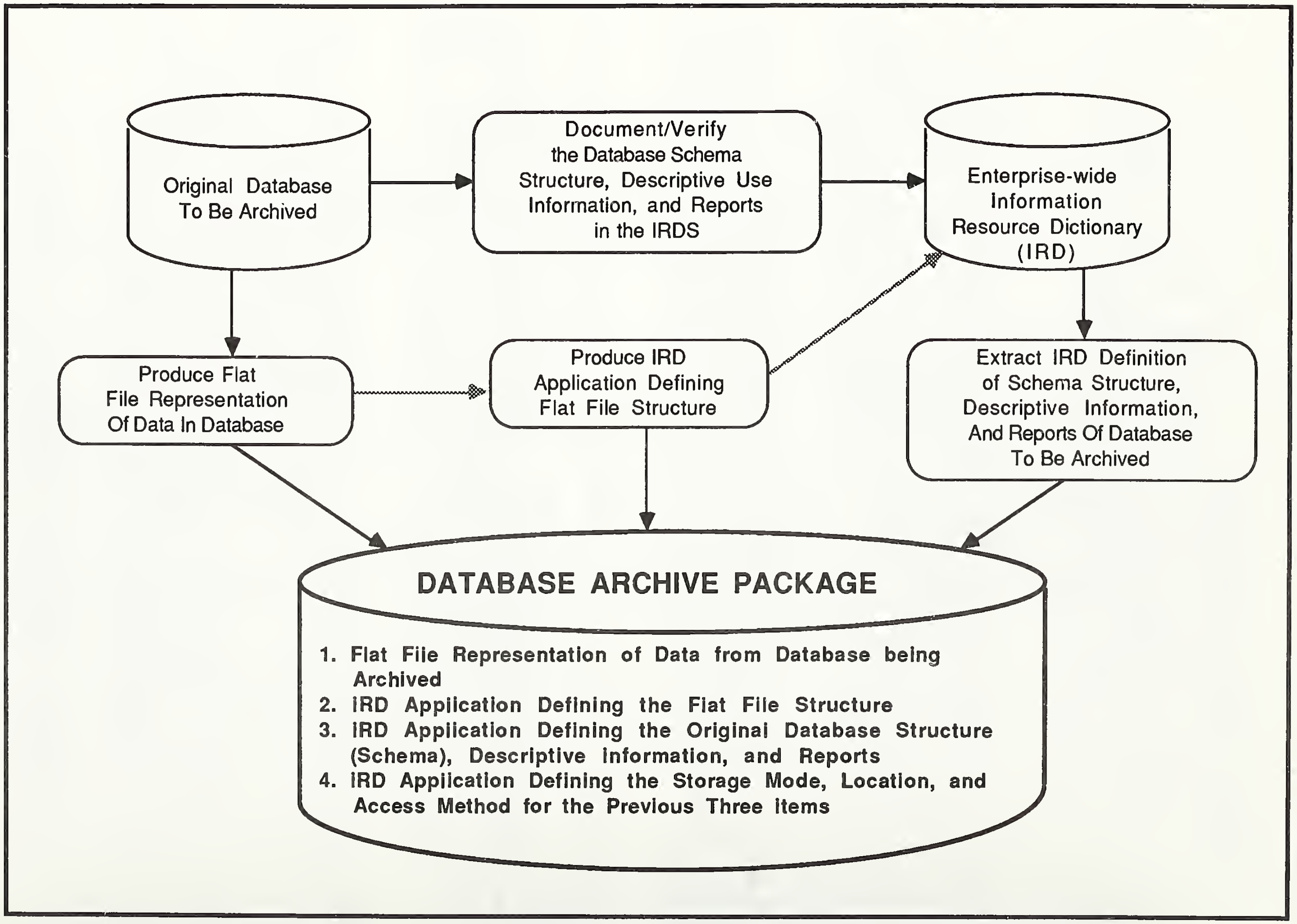

Original Database To Be Archived
Document/Verify the Database Schema Structure, Descriptive Use Information, and Reports in the IRDS
Enterprise-wide Information Resource Dictionary (IRD)
Produce Flat File Representation Of Data In Database
Produce IRD Application Defining Flat File Structure
Extract IRD Definition of Schema Structure, Descriptive Information, And Reports Of Database To Be Archived
DATABASE ARCHIVE PACKAGE
1. Flat File Representation of Data from Database being Archived
2. IRD Application Defining the Flat File Structure
3. IRD Application Defining the Original Database Structure (Schema), Descriptive Information, and Reports
4. IRD Application Defining the Storage Mode, Location, and Access Method for the Previous Three Items

FIGURE 15. DATABASE ARCHIVING MODEL

and its schema structures should be recorded in this IRD. (If all proper procedures were followed, this documentation of database schema and use should have taken place before the database became operational, and should have been placed in the Enterprise-wide IRD, from which it should now be extracted). This activity is indicated by the lozenge shape on the right. An export version of this IRD describing the database use, reports, and schema should be generated, using the IRDS Export/Import File Format. This IRD specification of the schema, descriptive information, and report structures, should be included in the archive package.

(4) A cross-referenced listing of the storage mode and storage location of these three archive package items should be maintained in another IRD application, that describes the how to use the entire database archive package. It is to this IRD that future researchers will first refer when trying to reconstitute the archived database.

As part of the generic archival function, Data Administration should ensure that the above four files remain linked together for each archived version of the database. The description of this linkage can be stored in an IRD application designed to keep track of archival material. By retaining the linkage between these four files, it should be possible to rebuild the original database under the same or, if necessary, a different database system.

Finally, Data Administration should ensure that appropriate multiple copies of archived databases are stored at remote sites, along with all needed documentation.

11. CONCLUSION

As discussed in this guide, it is vital to the success of Data Administration that the tools it utilizes to fulfill its various functions have the capability of automated interaction and interchange. Further, it is also clear that this interaction should be supported by tools compliant with the Information Resource Dictionary System (IRDS) Standard.

By using a variety of information analysis and management tools that access their information from the IRDS, acting as a central repository, information about an organization's information resources can be shared and integrated. By having all of the tools interact at this level, it is possible to use multiple types of tools that satisfy the various requirements of the organization, yet are still able to coordinate the different functions under a single mechanism.

Through the use of the single mechanism of the IRDS to coordinate and integrate information, as illustrated previously in figure 14, Data Administration can then successfully perform its data-driven functions to support the goals of the organization. Accordingly, an important requirement for any tool used by Data Administration is that it must be capable of interfacing with any compliant IRDS. Data Administration tools should be capable of interfacing with the IRDS via the IRDS Export/Import File Format or via the IRDS Services Interface, both of which are now in the process of being standardized.

REFERENCES

[CHAM89] Chambless, Linda C., Identification and Documentation of Objects Using Different Object-Oriented Requirements Analysis (OORA) Approach, Draft MITRE Technical Report, McLean, VA, 1989.

[CHEN76] Chen, Peter P.S., "The Entity-Relationship Model-Toward a Unified View of Data," ACM Transactions on Database Systems, 1:1, 9-36 (March 1976).

[DEMA79] DeMarco, Tom, Structured Analysis and System Specification, Yourdon Press, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979.

[FIPS76] Glossary for Computer Systems Security, FIPS PUB 39, National Institute of Standards and Technology, Gaithersburg, MD, February 1976.

[FIPS77] Guidelines on Evaluation of Techniques for Automated Personal Identification, FIPS PUB 48, National Institute of Standards and Technology, Gaithersburg, MD, April 1977.

[FIPS79] Guideline for Automatic Data Processing Risk Analysis, FIPS PUB 65, National Institute of Standards and Technology, Gaithersburg, MD, August 1979.

[FIPS80] Guidelines for Security of Computer Applications, FIPS PUB 73, National Institute of Standards and Technology, Gaithersburg, MD, June 1980.

[FIPS81] Guideline on Integrity Assurance and Control in Database Applications, FIPS PUB 88, National Institute of Standards and Technology, Gaithersburg, MD, August 1981.

[FIPS102] Guideline for Computer Security Certification and Accreditation, FIPS PUB 102, National Institute of Standards and Technology, Gaithersburg, MD, September 1983.

[FONG86] Fong, Elizabeth and Alan Goldfine, editors, Data Base Directions: Information Resource Management-- Making It Work, NBS Special Publication 500-139, National Institute of Standards and Technology, Gaithersburg, MD, June 1986.

[FONG88] Fong, Elizabeth and Bruce Rosen, Guide to Distributed Database Management, NBS Special Publication 500-154, National Institute of Standards and Technology, Gaithersburg, MD, April 1988.

[GOLD88] Goldfine, Alan, and Patricia Konig, A Technical Overview of the Information Resource Dictionary System (Second Edition), NBSIR 88-3700, National Institute of Standards and Technology, Gaithersburg, MD, January 1988.

[HAYK88] Haykin, Martha E., and Robert Warnar, Smart Cards and Computer Security, NBS Special Publication 500-157, National Institute of Standards and Technology, Gaithersburg, MD, 1988.

[KONI81] Konig, Patricia A., and Judith J. Newton, Federal Requirements for a Federal Information Processing Standard Data Dictionary System, NBSIR 81-2354, National Institute of Standards and Technology, Gaithersburg, MD, September 1981.

[JARD77] Jardine, Donald A., editor, The ANSI/SPARC DBMS Model Proceedings of the Second SHARE Working Conference on Data Base Management Systems, North-Holland Publishing Company (US distributor, American Elsevier Publishing Co. Inc.), 1977.

[LAW88] Law, Margaret Henderson, Guide to Information Resource Dictionary System Applications: General Concepts and Strategic Systems Planning, NBS Special Publication 500-152, National Institute of Standards and Technology, Gaithersburg, MD, April 1988.

[MART83] Martin, Roger J., and Wilma M. Osborne, Guidance on Software Maintenance, NBS Special Publication 500-106, National Institute of Standards and Technology, Gaithersburg, MD, December 1983.

[MILI70] Military Standard 490 (MIL-STD-490), Configuration Management Practices for Systems, Munitions, and Computer Programs, Department of Defense, December 31, 1970.

[NEWT87] Newton, Judith J., Guide on Data Entity Naming Conventions, NBS Special Publication 500-149, National Institute of Standards and Technology, Gaithersburg, MD, October 1987.

[NEUG85] Neugent, William, John Gilligan, Lance Hoffman, Lance Ruthberg, and Zella Ruthberg, Technology Assessment; Methods for Measuring the Level of Computer Security, NBS Special Publication 500-133, National Institute of Standards and Technology, Gaithersburg, MD, October 1985.

[OSBO89] Osborne, Wilma M., Software Configuration Management: An Overview, NIST Special Publication 500-161, National Institute of Standards and Technology, Gaithersburg, MD, March 1989.

[PAGE80] Page-Jones, Meilir, The Practical Guide to Structured Systems Design, Yourdon Press, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980.

[PERK84] Perkinson, Richard C., Data Analysis: The Key to Database Design, QED Information Sciences, Inc., Wellesley, MA, 1984.

[RUTH88] Ruthberg, Zella G., B.T. Fisher, W.E Perry, J.W. Lainhart, J.G. Cox, M. Gillen, and D.B. Hunt, Guide To Auditing For Controls and Security: A System Development Life Cycle Approach, NBS Special Publication 500-153, National Institute of Standards and Technology, Gaithersburg, MD, April 1988.

[STRO88] Stroustrup, Bjarne, "What is Object-Oriented Programming?" IEEE Software, May 1988.

[TEOR86] Teorey, Toby J., D. Yang and J.P. Fry, "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model," ACM Computing Surveys, Vol. 18, No. 2, June 1986.

[WARD89] Ward, Paul T., "How to Integrate Object Orientation with Structured Analysis and Design," IEEE Software, March 1989.

[WEBS83] Webster's Ninth New Collegiate Dictionary, Merriam Webster Inc., U.S.A., 1983.

[YOUR89] Yourdon, Edward, Modern Structured Analysis, Yourdon Press, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.

☆ U.S. GOVERNMENT PRINTING OFFICE: 1989-261-913-00076

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET *(See instructions)*	1. PUBLICATION OR REPORT NO. NIST/SP-500/173	2. Performing Organ. Report No.	3. Publication Date October 1989

4. TITLE AND SUBTITLE

Guide to Data Administration

5. AUTHOR(S)

Bruce K. Rosen and Margaret H. Law

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
(formerly NATIONAL BUREAU OF STANDARDS)
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

NIST SP

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Information Systems Engineering Division
National Computer Systems Laboratory
National Institute of Standards and Technology

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 89-600766

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or l[illegible] significant*

Data Administration is the management of information describing the data, functions, operations, and structure of automatic data processing (ADP) systems and databases. Data Administration collects descriptive information about an organization's data, functions, and operations to support the development of the ADP systems, to provide full system documentation during systems operation and maintenance, and to support redesign efforts required as the needs of an organization evolve.

This guide provides a reference model for the various activities performed by Information Resource Management, Data Administration, Data Modeling Tools Administration, and Database Administration. The functions of Data Administration are discussed in detail. Data Administration is responsible for defining an information architecture, and for establishing policies for naming conventions, information modeling techniques and methodologies, data element specification, system information integration, and data protection. The importance of the Three Schema Architecture is described.

Computing tools useful for Data Administration, such as data dictionary systems and computer-aided software engineering (CASE) tools, are addressed. The guide stresses the useful features of these computing tools. Use of the Information Resource Dictionary System (IRDS) Standard, approved by the American National Standards Institute (ANSI) and a Federal Information Processing Standard (FIPS), is discussed.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

data administration; data management; data modeling; information resource dictionary system; IRDS; information resource management; IRM

13. AVAILABILITY

[X] Unlimited

☐ For Official Distribution. Do Not Release to NTIS

[X] Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

81

15. Price

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SYSTEMS TECHNOLOGY

Superintendent of Documents
Government Printing Office
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Institute of Standards and Technology Special Publication 500-.

Name ______________________________

Company ______________________________

Address ______________________________

City ______________ State ______________ Zip Code ______________

(Notification key N-503)